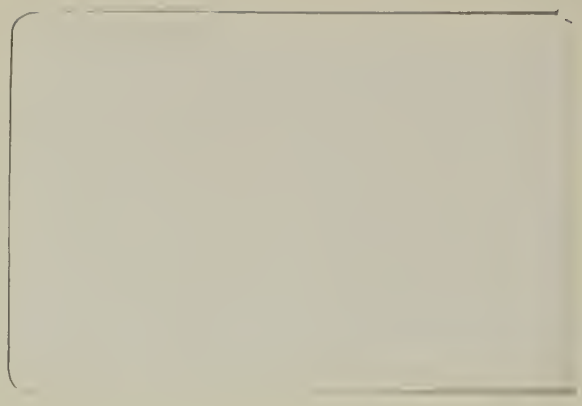
HIGHWAY INFORMATION SYSTEM
VOLUME 2:
PROGRAMMER INFORMATION
PART II

Department of Civil Engineering & Engineering Mechanics Montana State University Bozeman



HIGHWAY INFORMATION SYSTEM
VOLUME 2:
PROGRAMMER INFORMATION
PART II

Prepared for the

STATE OF MONTANA
DEPARTMENT OF HIGHWAYS
PLANNING AND RESEARCH BUREAU

In cooperation with the

U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL HIGHWAY ADMINISTRATION

Prepared by

Glen L. Martin, Larry J. Coats, and Michael J. Meldahl

DEPARTMENT OF CIVIL ENGINEERING AND ENGINEERING MECHANICS
MONTANA STATE UNIVERSITY
Bozeman, Montana 59715

June, 1972

FOREWORD

This report is a documentation of the highway data bank research undertaken by the Department of Civil Engineering and Engineering Mechanics, Montana State University. The research was sponsored by the Montana Department of Highways in cooperation with the U.S. Department of Transportation, Federal Highway Administration.

Conceptually, the CE & EM Department was responsible for developing an information retrieval system for rapid access to highway data. Specifically, the responsibility was to produce the Roadlog, Traffic by Sections, Accident by Sections and Sufficiency by Sections reports as a direct application of the system. In addition, preliminary investigation of the feasiblity of a geometrics file and a preliminary investigation of the storage and retrieval of visual images was included in the project objectives.

In light of the foregoing, it is desirable to present the report in two volumes: Highway Information System Volume 1: User Information, and Highway Information System Volume 2: Programmer Information. Volume 1 deals with the use of the system, including information on data coding and on the execution of programs within the system. Volume 2 deals with the detailed operation of the system, providing information on the modification of programs existing within the system as well as on the addition of programs to the system. Volume 1 is a prerequisite publication to Volume 2.

In developing the system, the CE & EM Department has had the privilege of using an IBM OS 360/40 computer located at the Data Processing Bureau of the Montana Department of Highways in Helena. PL/I has been used as the programming language for nearly all of the HIS routines because of its versatility in input-output (I/O) and interchangeability of files. BAL (assembler) has been used for several routines because of its increased capabilities and efficiency over other languages.

The project could never have progressed to its current state were it not for the continual encouragement from and the patient, sustained assistance of both the Planning and Research Bureau and the Data Processing Bureau of the Montana Department of Highways, and of the Montana State Highway Patrol.

The project conclusion was also hastened by the significant effort of other project personnel: Francis C. F. Yu, Leroy R. Zook, Philip A. House,

Alfred C. Scheer, Paul W. Burkhart, Robert C. Smith, Harry E. Hughes, Ronald E. Billstein, Daniel D. Urbach and Donald R. Reichmuth. Their assistance has been invaluable.

HIGHWAY INFORMATION SYSTEM
VOLUME 2: PROGRAMMER INFORMATION

TABLE OF CONTENTS

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

LIST OF FIGURES

LIST OF TABLES

CHAPTER 2-IV

ACCIDENT PROGRAMMER INFORMATION

Introduction

This chapter presents a description of the programs comprising the Accident subsystem of HIS (Highway Information System). It is designed for utilization with the publication Highway Information System Volume 1: User Information.

Accident Detail File Description

Data Set Name	HIS.ACCIDENT
Organization	Indexed Sequential
Logical Record Length	96
Physical Record Length	1632
Key Length	12
Volume Serial Number	231432

The internal format of an Accident Detail record is shown in PL/I terminology in Figure 2-IV-1. Most of the numeric fields are stored in packed decimal format to conserve storage and improve efficiency.

Two additional fields are added to the record: a "reportable" field containing an "X" if there were any injuries or fatalities or if damage of over $250 occurred to one or more vehicles, and an "investigated" field containing an "X" if the accident was investigated.

Accident Vehicle File Description

Data Set Name	HIS.ACCVEH
Organization	Indexed Sequential
Logical Record Length	136
Physical Record Length	1632
Key Length	15
Volume Serial Number	231415

```
1   DETAIL_RECORD,
    2  DELETE_CHARACTER                    CHAR(1),
    2  KEY                                 CHAR(12),
    2  DATE_&_TIME_OCCURRED,
       3  (MONTH,DAY,YEAR,HOUR,MIN)        DEC FIXED (2,0),
    2  DATE_&_TIME_NOTIFIED
       3  (MONTH,DAY,YEAR,HOUR,MIN)        DEC FIXED (2,0),
    2  DATE_&_TIME_ARRIVED
       3  (MONTH,DAY,YEAR,HOUR,MIN)        DEC FIXED (2,0),
    2  CITY_NUMBER                         DEC FIXED (3,0),
    2  COUNTY_NUMBER                       DEC FIXED (2,0),
    2  MILEPOST                            CHAR(12),
    2  FIRST_HARMFUL_EVENT                 DEC FIXED (2,0),
    2  FIRST_OBJECT_HIT                    DEC FIXED (2,0),
    2  INJURY_SEVERITY                     DEC FIXED (1,0),
    2  DAMAGE_SEVERITY                     DEC FIXED (1,0),
    2  CLASS_OF_TRAFFICWAY                 DEC FIXED (1,0),
    2  ROADWAY_RELATED_LOCATION            DEC FIXED (1,0),
    2  JUNCTION_RELATED_LOCATION           DEC FIXED (1,0),
    2  (#_VEHICLES,#_PEDESTRIANS)          DEC FIXED (2,0),
    2  (#_FATALITIES,#_INJURIES)           DEC FIXED (2,0),
    2  WEATHER_CONDITION                   DEC FIXED (1,0),
    2  ROAD_CONDITION                      DEC FIXED (1,0),
    2  LIGHT_CONDITION                     DEC FIXED (1,0),
    2  TRAFFIC_CONTROLS                    DEC FIXED (2,0),
    2  OTHER_DAMAGE,
       3  TYPE                             DEC FIXED (2,0),
       3  SEVERITY                         DEC FIXED (1,0),
       3  OWNER                            DEC FIXED (1,0),
    2  POSTED_SPEED                        DEC FIXED (2,0),
    2  ENGINEERING_STUDY                   CHAR(1),
    2  CONTRIBUTING_CIRCUMSTANCES(2)       DEC FIXED (2,0),
    2  COLLISION_TYPE                      DEC FIXED (1,0),
    2  REPORTABLE                          CHAR(1),
    2  INVESTIGATED                        CHAR(1),
    2  DUMMY                               CHAR(1);
```

Figure 2-IV-1. Accident detail file structure.

The internal format of an Accident Vehicle record is shown in PL/I terminology in Figure 2-IV-2. The TYPE_CODE field of the key contains an "A" on records describing vehicles, a "B" on records describing pedestrians, and a "C" on extra records defining additional injuries to persons in one of the vehicles. The SEQUENCE_NUMBER field for each accident begins with 01, and increases in increments of one.

Accident Directory File Description

Data Set Name	HIS.ACCDIRI
Organization	Indexed Sequential
Logical Record Length	44
Physical Record Length	748
Key Length	25
Volume Serial Number	231415

The Accident Directory file record is shown in PL/I terminology in Figure 2-IV-3. All of the data items required from the Accident Detail file for the Accident by Sections report are copied into the Directory file as it is built, saving later access to the Detail file.

Accident Report File Description

Data Set Name	HIS.ACCSECT
Organization	Indexed Sequential
Logical Record Length	104
Physical Record Length	1040
Key Length	13
Volume Serial Number	231415

The Accident Report file record structure is shown in Figure 2-IV-4. The various data items in this file are derived from the Roadlog, Traffic, True Mileage, and Accident Directory files.

```
1   VEHICLE_RECORD,
    2  DELETE_CHARACTER                    CHAR(1),
    2  KEY,
       3  ACCIDENT_NUMBER                  CHAR(12),
       3  TYPE_CODE                        CHAR(1),
       3  SEQUENCE_NUMBER                  CHAR(2),
    2  LAST_NAME                           CHAR(20),
    2  INITIALS(2)                         CHAR(1),
    2  DRIVERS_LICENSE                     CHAR(17),
    2  STATE                               CHAR(2),
    2  BIRTHDAY                            CHAR(6),
    2  RE_EXAM                             CHAR(1),
    2  CHARGE_CODE                         CHAR(6),
    2  SUMMONS_NUMBER                      CHAR(6),
    2  CONTRIBUTING_CIRCUMSTANCES(5)       DEX FIXED (1,0),
    2  DRIVER,
       3  ALCOHOL                          DEC FIXED (1,0),
       3  SEX                              CHAR(1),
       3  INJURY                           DEC FIXED (1,0),
       3  AGE                              DEC FIXED (2,0),
    2  FRONT_CENTER                        LIKE DRIVER,
    2  FRONT_RIGHT                         LIKE DRIVER,
    2  REAR_LEFT                           LIKE DRIVER,
    2  REAR_CENTER                         LIKE DRIVER,
    2  REAR_RIGHT                          LIKE DRIVER,
    2  VEHICLE_YEAR                        DEC FIXED (2,0),
    2  INTENT                              DEC FIXED (2,0),
    2  VEHICLE_BODY                        DEC FIXED (2,0),
    2  TRAILER                             DEC FIXED (1,0),
    2  INTERSTATE_TRAFFIC                  CHAR(1),
    2  VEHICLE_ID_OR_LICENSE               CHAR(15),
    2  DAMAGE_GREATER_THAN_250             CHAR(1),
    2  DAMAGE_SEVERITY                     DEC FIXED (1,0);
```

Figure 2-IV-2. Accident vehicle file structure.

```
1   DIRECTORY_RECORD
    2  DELETE_CHARACTER                    CHAR(1),
    2  KEY,
       3  ROUTE_SYSTEM                     CHAR(1),
       3  ROUTE_NUMBER                     CHAR(3),
       3  REFERENCE_POST                   CHAR(3),
       3  DISTANCE                         CHAR(6),
       3  ACCIDENT_NUMBER                  CHAR(12),
    2  #_FATALITIES                        DEC FIXED (2,0),
    2  #_INJURIES                          DEC FIXED (2,0),
    2  DATE,
       3  MONTH                            DEC FIXED (2,0),
       3  DAY                              DEC FIXED (2,0),
       3  YEAR                             DEC FIXED (2,0),
    2  HOUR                                DEC FIXED (2,0),
    2  FIRST_HARMFUL_EVENT                 DEC FIXED (2,0),
    2  COLLISION_TYPE                      DEC FIXED (1,0),
    2  ROAD_SURFACE_CONDITION              DEC FIXED (1,0),
    2  #_LANES                             DEC FIXED (1,0),
    2  DATE_FLAG                           CHAR(1);
```

Figure 2-IV-3. Accident directory file structure.

```
1   REPORT_RECORD,
    2  DELETE_CHARACTER                    CHAR(1),
    2  KEY,
       3  ROUTE_SYSTEM                     CHAR(1),
       3  ROUTE_NUMBER                     CHAR(3),
       3  REFERENCE_POST                   CHAR(3),
       3  DISTANCE                         CHAR(6),
    2  REMARK                              CHAR(1),
    2  DESCRIPTION                         CHAR(35),
    2  SECTION_LENGTH                      DEC FIXED (7,3),
    2  FIRST_YEAR,
       3  YEAR                             DEC FIXED (2,0),
       3  AVERAGE_DAILY_TRAFFIC            DEC FIXED (5,0),
       3  #_ACCIDENTS                      DEC FIXED (3,0),
       3  #_INJURY_ACCIDENTS               DEC FIXED (3,0),
       3  #_FATAL_ACCIDENTS                DEC FIXED (3,0),
       3  #_INJURIES                       DEC FIXED (3,0),
       3  #_FATALITIES                     DEC FIXED (3,0),
    2  SECOND_YEAR                         LIKE FIRST_YEAR,
    2  THIRD_YEAR                          LIKE FIRST_YEAR,
    2  NUMBER_OF_LANES                     DEC FIXED (1,0),
    2  CITY_NUMBER                         DEC FIXED (3,0),
    2  DUMMY                               CHAR(2);
```

Figure 2-IV-4. Accident report file structure.

Accident Limits File Description

Data Set Name	HIS.ACCLIM
Organization	Indexed Sequential
Logical Record Length	24
Physical Record Length	480
Key Length	4
Volume Serial Number	231415

The format of an Accident Limits record is shown in Figure 2-IV-5. The Limits file is generated from the Accident Report file as the final step prior to printing the Accident by Sections report.

Subroutines

Several of the Accident programs utilize subroutines. These subroutines are stored in object module format in cataloged library HIS.OBJECT. The subroutines are:

CONVACC --

Object Module Name	CONVACC
Language	PL/I
Files	SYSPRINT
Entry Points	CONVABR CONVRAB CONVCDR CONVRCD

CONVACC is used to convert accident data cards into the internal file format. The entry points into the routine are:

CONVABR	converts an A-B (or E-F) card sequence into a detail record.
CONVRAB	converts a detail record into an A-B (or E-F) card sequence.

```
1   LIMITS_RECORD,
    2  DELETE_CHARACTER                  CHAR(1),
    2  KEY,
       3  ROUTE_SYSTEM                   CHAR(1),
       3  ROUTE_NUMBER                   CHAR(3),
    2  FIRST_YEAR,
       3  LOWER_LIMIT                    DEC FIXED (5,3),
       3  UPPER_LIMIT                    DEC FIXED (5,3),
    2  SECOND_YEAR                       LIKE FIRST_YEAR,
    2  THIRD_YEAR                        LIKE FIRST_YEAR,
    2  DUMMY                             CHAR(1);
```

Figure 2-IV-5. Accident limits file structure.

CONVCDR converts a C-D (or G-H) card sequence into a vehicle record.

CONVRCD converts a vehicle record into a C-D (or G-H) card sequence.

When calling CONVABR or CONVRAB, an eighty-byte string is passed for each card (an A card and a B card), a 96-byte string for the detail record, and a statement label to which control will be passed in the event of an error in conversion. When calling CONVCDR or CONVRCD, two 80-byte strings (the C and D cards), a 136-byte string (the vehicle record), a 96-byte string (the already-converted detail record), and an error-return statement label are passed. An example of CONVACC usage is:

```
      DECLARE (A,B,C,D) CHAR(80), DET CHAR(96), VEH CHAR(120);
      DECLARE (#_VEH,#_PED) PIC'ZZ';
      GET FILE (SYSIN) EDIT (A,B) (A(80));
      CALL CONVABR (A,B,DET,ERROR);
      #_VEH=SUBSTR(A,50,2);
      #_PED = SUBSTR(A,52,2);
      DO I=1 TO #_VEH + #_PED;
         GET FILE (SYSIN) EDIT (C,D) (A(80));
         CALL CONVCDR (C,D,VEH,DET,ERROR);
         END;
          .
          .
          .
ERROR:
          .
          .
          .
```

The CONVACC program listing follows:

```
CONVERT:  PROCEDURE;

  1: CONVERT:  PROCEDURE;

  2: /* PARAMETERS */
  3: DECLARE
  4:   (A_CARD,B_CARD,C_CARD,D_CARD) CHAR(80),
  5:    DET_RECORD CHAR(96),
  6:    VEH_RECORD CHAR(136),
  7:    ERROR_RETURN LABEL;

  8: /* DETAIL RECORD */
  9: DECLARE
 10:    1  DET BASED (PTR_DET),
 11:       2  DUM1 CHAR(1),
 12:       2  KEY CHAR(12),
 13:       2  OCCURRED,
 14:          3 (MONTH,DAY,YEAR,HOUR,MIN) DEC FIXED (3,0),
 15:       2 (NOTIFIED,ARRIVED) LIKE DET.OCCURRED,
 16:       2 (CITY_#,CNTY_#) DEC FIXED (3,0),
 17:       2  MILEPOST CHAR(12),
 18:       2  BLOCK_A,
 19:          3 (FIRST_EVENT,FIRST_OBJ) DEC FIXED (3,0),
 20:          3 (INJ_SEV,DAM_SEV,TRAFFICWAY,RDY_REL,JCT_REL)
 21:             DEC FIXED (1,0),
 22:       2  BLOCK_B,
 23:          3 (#_VEH,#_PED,#_FAT,#_INJ) DEC FIXED (3,0),
 24:          3 (WEATHER,ROAD,LIGHT) DEC FIXED (1,0),
 25:       2  BLOCK_C,
 26:          3 (CONTROLS,OTH_DAM_TYPE) DEC FIXED (3,0),
 27:          3 (OTH_DAM_SEV,OTH_DAM_OWNER) DEC FIXED (1,0),
 28:          3  SPEED DEC FIXED (3,0),
 29:          3  ENG_STUDY CHAR(1),
 30:          3  ANAL(2) DEC FIXED (3,0),
 31:          3  COLL_TYPE DEC FIXED (1,0),
 32:       2 (REPORTABLE,INVESTIGATED) CHAR(1),
 33:       2  DUM2 CHAR(1);

 34: /* VEHICLE/PEDESTRIAN RECORD */
 35: DECLARE
 36:    1  VEH BASED (PTR_VEH),
 37:       2  DUM1 CHAR(1),
 38:       2  KEY CHAR(12),
 39:       2  VEH_PED CHAR(1),
 40:       2  VEH_# PIC'99',
 41:       2  LAST_NAME CHAR(22),
 42:       2  DRIV_LICENSE CHAR(17),
 43:       2  STATE CHAR(2),
 44:       2  BIRTHDAY CHAR(6),
 45:       2  RE_EXAM CHAR(1),
 46:       2 (CHARGE,SUMMONS) CHAR(6),
 47:       2  CONTR_CIRC(5) DEC FIXED (1,0),
 48:       2  PASS(6),
 49:          3  ALCOHOL DEC FIXED (1,0),
 50:          3  SEX CHAR(1),
 51:          3  INJ DEC FIXED (1,0),
 52:          3  AGE DEC FIXED (3,0),
 53:       2 (VEH_YEAR,INTENT,BODY) DEC FIXED (3,0),
 54:       2  TRAILER DEC FIXED (1,0),
 55:       2  INTERSTATE_TRAF CHAR(1),
```

```
CONVERT:  PROCEDURE;

 56:        2  VEH_ID CHAR(15),
 57:        2  REPORTABLE CHAR(1),
 58:        2  VEH_DAM DEC FIXED (1,0);

 59: /* "A" CARD */
 60: DECLARE
 61:     1  A BASED (PTR_A),
 62:        2  CODE CHAR(1),
 63:        2  SEQ PIC'Z',
 64:        2  KEY CHAR(12),
 65:        2  OCCURRED,
 66:           3 (MONTH,DAY,YEAR,HOUR,MIN) PIC'ZZ',
 67:        2  CITY_# PIC'ZZZ',
 68:        2  CNTY_# PIC'ZZ',
 69:        2  MILEPOST CHAR(12),
 70:        2  BLOCK_A,
 71:           3 (FIRST_EVENT,FIRST_OBJ) PIC'ZZ',
 72:           3 (INJ_SEV,DAM_SEV,TRAFFICWAY,RDY_REL,JCT_REL)
 73:              PIC'Z',
 74:        2  BLOCK_B,
 75:           3 (#_VEH,#_PED,#_FAT,#_INJ) PIC'ZZ',
 76:           3 (WEATHER,ROAD,LIGHT) PIC'Z',
 77:        2  BLOCK_C,
 78:           3 (CONTROLS,OTH_DAM_TYPE) PIC'ZZ',
 79:           3 (OTH_DAM_SEV,OTH_DAM_OWNER) PIC'Z',
 80:           3  SPEED PIC'ZZ',
 81:           3  ENG_STUDY CHAR(1),
 82:           3  ANAL(2) PIC'ZZ',
 83:           3  COLL_TYPE PIC'Z';

 84: /* "B" CARD */
 85: DECLARE
 86:     1  B BASED (PTR_B),
 87:        2  CODE CHAR(1),
 88:        2  SEQ PIC'Z',
 89:        2  KEY CHAR(12),
 90:        2  NOTIFIED,
 91:           3 (MONTH,DAY,YEAR,HOUR,MIN) PIC'ZZ',
 92:        2  ARRIVED LIKE B.NOTIFIED;

 93: /* "C" CARD */
 94: DECLARE
 95:     1  C BASED (PTR_C),
 96:        2  CODE CHAR(1),
 97:        2  SEQ PIC'Z',
 98:        2  KEY CHAR(12),
 99:        2  LAST_NAME CHAR(22),
100:        2  DRIV_LICENSE CHAR(17),
101:        2  STATE CHAR(2),
102:        2  BIRTHDAY CHAR(6),
103:        2  RE_EXAM CHAR(1),
104:        2 (CHARGE,SUMMONS) CHAR(6);

105: /* "D" CARD */
106: DECLARE
107:     1  D BASED (PTR_D),
108:        2  CODE CHAR(1),
109:        2  SEQ PIC'Z',
```

```
CONVERT:  PROCEDURE;

110:        2  KEY CHAR(12),
111:        2  CONTR_CIRC(5) PIC'Z',
112:        2  PASS(6),
113:           3  ALCOHOL PIC'Z',
114:           3  AGE PIC'ZZ',
115:           3  SEX CHAR(1),
116:           3  INJ PIC'Z',
117:        2 (VEH_NO,VEH_INTENT,PED_NO,PED_INTENT) PIC'ZZ',
118:        2  BODY PIC'ZZ',
119:        2  TRAILER PIC'Z',
120:        2  VEH_YEAR PIC'ZZ',
121:        2  INTERSTATE_TRAF CHAR(1),
122:        2  VEH_ID CHAR(15),
123:        2  REPORTABLE CHAR(1),
124:        2  VEH_DAM PIC'Z';

125: /***** ENTRY TO CONVERT A-B SEQUENCE TO DETAIL RECORD *****/

126: CONVABR:  ENTRY (A_CARD,B_CARD,DET_RECORD,ERROR_RETURN);

127:     ON ERROR GOTO ERROR_RETURN;
128:     DET_RECORD = ' ';
129:     PTR_DET = ADDR(DET_RECORD);
130:     PTR_A = ADDR(A_CARD);
131:     PTR_B = ADDR(B_CARD);
132:     DET = A, BY NAME;
133:     DET = B, BY NAME;
134:     IF DET.INJ_SEV¬=0 THEN DET.REPORTABLE = 'X';
135:     IF A.CODE='E' THEN DET.INVESTIGATED = 'X';
136:     RETURN;


137: /***** ENTRY TO CONVERT DETAIL RECORD TO A-B SEQUENCE *****/

138: CONVRAB:  ENTRY (A_CARD,B_CARD,DET_RECORD,ERROR_RETURN);

139:     ON ERROR GOTO ERROR_RETURN;
140:     PTR_DET = ADDR(DET_RECORD);
141:     PTR_A = ADDR(A_CARD);
142:     PTR_B = ADDR(B_CARD);
143:     IF DET.INVESTIGATED='X' THEN DO;
144:        A_CARD = 'E0';
145:        B_CARD = 'F0';
146:        END;
147:     ELSE DO;
148:        A_CARD = 'A0';
149:        B_CARD = 'B0';
150:        END;
151:     A = DET, BY NAME;
152:     B = DET, BY NAME;
153:     RETURN;


154: /***** ENTRY TO CONVERT C-D SEQUENCE TO VEHICLE RECORD *****/

155: CONVCDR:  ENTRY (C_CARD,D_CARD,VEH_RECORD,DET_RECORD,ERROR_RETURN);

156:     ON ERROR GOTO ERROR_RETURN;
```

```
CONVERT:  PROCEDURE;

157:     VEH_RECORD = ' ';
158:     PTR_VEH = ADDR(VEH_RECORD);
159:     PTR_DET = ADDR(DET_RECORD);
160:     PTR_C = ADDR(C_CARD);
161:     PTR_D = ADDR(D_CARD);
162:     IF D.CODE¬='I' THEN VEH = C, BY NAME;
163:     VEH = D, BY NAME;
164:     IF D.VEH_NO¬=0 THEN DO;
165:        VEH.VEH_PED = 'A';
166:        VEH.VEH_# = D.VEH_NO;
167:        VEH.INTENT = D.VEH_INTENT;
168:        END;
169:     ELSE IF D.PED_NO¬=0 THEN DO;
170:        VEH.VEH_PED = 'B';
171:        VEH.VEH_# = D.PED_NO;
172:        VEH.INTENT = D.PED_INTENT;
173:        END;
174:     ELSE VEH.VEH_PED = 'C';
175:     IF VEH.REPORTABLE='X' THEN DET.REPORTABLE = 'X';
176:     RETURN;


177: /***** ENTRY TO CONVERT VEHICLE RECORD TO C-D SEQUENCE *****/

178: CONVRCD:  ENTRY (C_CARD,D_CARD,VEH_RECORD,DET_RECORD,ERROR_RETURN);

179:     ON ERROR GOTO ERROR_RETURN;
180:     PTR_VEH = ADDR(VEH_RECORD);
181:     PTR_DET = ADDR(DET_RECORD);
182:     PTR_C = ADDR(C_CARD);
183:     PTR_D = ADDR(D_CARD);
184:     IF VEH.VEH_PED='C' THEN D.CODE = 'I';
185:     ELSE IF DET.INVESTIGATED='X' THEN DO;
186:        C_CARD = 'G';
187:        D_CARD = 'H';
188:        END;
189:     ELSE DO;
190:        C_CARD = 'C';
191:        D_CARD = 'D';
192:        END;
193:     C = VEH, BY NAME;
194:     D = VEH, BY NAME;
195:     IF VEH.VEH_PED='A' THEN DO;
196:        D.VEH_NO = VEH.VEH_#;
197:        D.VEH_INTENT = VEH.INTENT;
198:        C.SEQ, D.SEQ = VEH.VEH_#;
199:        END;
200:     ELSE IF VEH.VEH_PED='B' THEN DO;
201:        D.PED_NO = VEH.VEH_#;
202:        D.PED_INTENT = VEH.INTENT;
203:        C.SEQ, D.SEQ = VEH.VEH_# + DET.#_VEH;
204:        END;
205:     RETURN;

206: END CONVERT;
```

GETDAY --

```
Object Module Name . . . . . GETDAY
Language . . . . . PL/I
Files . . . . . SYSPRINT
Entry Point . . . . . GETDAY
```

The month, day, and year (20th century) are passed to the subroutine. The julian day and the day of the week are returned. All items are decimal fixed (3,0). The day is returned as a number between 1 and 7:

```
1  Saturday
2  Sunday
3  Monday
4  Tuesday
5  Wednesday
6  Thursday
7  Friday
```

An example of GETDAY usage is:

```
TEST:  PROC OPTIONS (MAIN);
DECLARE (MONTH,DAY,YEAR,JULIAN,DAY_OF_WEEK) DEC FIXED (3,0);
MONTH = 1;  /* JANUARY */
DAY = 1;
YEAR = 72;
CALL GETDAY (MONTH,DAY,YEAR,JULIAN,DAY_OF_WEEK);
END TEST;
```

The julian day (1) and the day of the week (1, Saturday) are returned.

The GETDAY program listing follows:

```
GETDAY:  PROCEDURE (MM,DD,YY,JULIAN,DAY);

  1: GETDAY:  PROCEDURE (MM,DD,YY,JULIAN,DAY);
  2: DECLARE
  3:   (MM,DD,YY,JULIAN,DAY) DEC FIXED (3,0),
  4:   (I,J,YEAR) DEC FIXED (5,0),
  5:    FAC(12) DEC FIXED (3,0) STATIC INIT
  6:       (0,31,59,90,120,151,181,212,243,273,304,334);
  7:
  8:    JULIAN = FAC(MM) + DD;
  9:    I = YY/4;
 10:    I = I*4;
 11:    IF I=YY & MM>2 THEN JULIAN = JULIAN + 1;
 12:    YEAR = 1900 + YY;
 13:    I = YEAR - 1;
 14:    J = JULIAN + YEAR + I/4 + I/400 - I/100;
 15:    I = J/7;
 16:    I = I*7;
 17:    DAY = (J-I) + 1;
 18:    END GETDAY;
```

Program Descriptions

Each program in the accident subsystem is stored in load module format in cataloged library HIS.LOADLIB, from which it is retrieved for execution by the HIS supervisor when requested. The member name for each program is given with the program description.

This section of the manual presents a write-up on each program in the Accident subsystem. An attempt has been made in the source listing itself to document the program by means of appropriate variable names and comments.

EDIT-DATA-CARDS --

```
Member Name . . . . . PYA
Language . . . . . PL/I
Subroutines . . . . . PRINTX1
Files . . . . . SYSPRINT -- IBM messages
                        PRINTER  -- PYA output
                        EDITIN   -- Input data cards
                        EDITOUT  -- Output file
Instruction . . . . . 1 - 3  "PYA"
```

EDIT-DATA-CARDS performs pre-loading data testing functions on the accidents punched each month to help insure valid data in the accident files. The various tests, and the corresponding error messages, are outlined in the publication Highway Information System Volume 1: User Information.

Each accident consists of an "A" card, an optional "B" card, a "C"-"D" card sequence for each vehicle and pedestrian, and optional "I" cards for showing additional injuries.

The algorithm utilized by EDIT-DATA-CARDS and LOAD-DATA-CARDS consists of reading all of the accident cards prior to processing the "C" and "D" cards. Each time a "C" card is read, it is placed into an array of "C" cards. Each "D" card is placed into an array of "D" cards. Each "I" card is placed into an "I" array. The number of each type read is stored. If an unequal number of "C" and "D" cards are read, an error message is printed. Otherwise, the "C" and "D" sequences are matched by position within the array. Serious errors (such as a missing "A" card or an unmatched "C" or "D" card)

cause immediate rejection. When other errors are detected, a message is printed immediately, and a flag (ERR_FLAG) is set to indicate an error. Hence, all of the remaining checks may still be made, and as many messages as required printed.

The PYA program listing follows:

```
/* ACCIDENT FILE EDIT PHASE */

 1: /* ACCIDENT FILE EDIT PHASE */

 2: EDITOR:  PROCEDURE (PARM) OPTIONS (MAIN);

 3: /* INSTRUCTION AND PRINT ROUTINE */
 4: DECLARE
 5:    PARM CHAR(100),
 6:    INSTR CHAR(80) EXT,
 7:    DDNAME CHAR(8) DEF INSTR POS(24),
 8:    #_HDGS PIC'Z' DEF INSTR POS(72),
 9:   (PRINTER, HEADING(9)) CHAR(132) EXT,
10:    PRINTXA ENTRY (PIC'Z',PIC'ZZ'),
11:    PRINTX ENTRY (PIC'Z');

12: /* DATA INPUT */
13: DECLARE
14:    CODE CHAR(1) BASED (PTR_DATA),
15:    CARD CHAR(80) BASED (PTR_DATA),
16:    DATA FILE RECORD;

17: /* DATA OUTPUT */
18: DECLARE
19:    OUT CHAR(81),
20:    EDIT FILE RECORD OUTPUT ENV (F(3483,81));

21: /* "A" CARD */
22: DECLARE
23:    S_A CHAR(80) STATIC,
24:    1  A DEF S_A,
25:       2 (CODE,SEQ) CHAR(1),
26:       2  KEY CHAR(12),
27:       2  DUM2 CHAR(4),
28:       2  YEAR CHAR(2),
29:       2  DUM3 CHAR(4),
30:       2  CITY_# PIC'999',
31:       2  CNTY_# CHAR(2),
32:       2  SYS_CODE CHAR(1),
33:       2  RT_# CHAR(3),
34:       2  MILEPOST CHAR(8),
35:       2  DUM4 CHAR(4),
36:       2 (INJ_SEV,DAM_SEV) CHAR(1),
37:       2  TRAF CHAR(1),
38:       2  DUM5 CHAR(2),
39:       2 (#_VEH,#_PED,#_FAT,#_INJ) PIC'99';

40: /* "B" CARD */
41: DECLARE
42:    S_B CHAR(80) STATIC;

43: /* "C" AND "D" CARDS */
44: DECLARE
45:   (S_C(50),S_D(50),S_I(20)) CHAR(80) STATIC;

46: /* CITY TABLE */
47: DECLARE
48:    1  CITY BASED (PTR_TBL),
49:       2  DUM1 CHAR(59),
50:       2  CNTY_# CHAR(2),
```

```
/* ACCIDENT FILE EDIT PHASE */

51:     CNTY(0:126) CHAR(2),
52:     TABLE FILE INT RECORD;

53: /* OTHER VARIABLES */
54: DECLARE
55:    (DEC_#_VEH, DEC_#_PED, #_VEH, #_PED, #_FAT, #_INJ) DEC FIXED (3,0
56:    (INJ_SEV,DAM_SEV) CHAR(1),
57:     Z1 PIC'Z',
58:     Z2 CHAR(1) DEF Z1,
59:     ACC_# CHAR(12),
60:     ACC_#_SEQ PIC'99' DEF ACC_# POS(11),
61:     ACC_NOS(2000) CHAR(12),
62:     SAVE_ACC_# CHAR(14),
63:     DUP_ACC CHAR(1),
64:     #_ACC DEC FIXED (5,0),
65:     COR_FLAG CHAR(1),
66:    (ACC_CNTR, ERR_CNTR) DEC FIXED (7,0),
67:    (#_C,#_D) DEC FIXED (3,0),
68:     #_I DEC FIXED (3,0),
69:     CHR(80) CHAR(1) BASED (PTR),
70:     B_FLAG CHAR(1),
71:    (ERR_FLAG, END_FLAG) CHAR(1),
72:     BLANKS CHAR(132) STATIC INIT (' '),
73:    (I,J) DEC FIXED (7,0);


74: /***** INITIALIZATION *****/

75:     CALL INIT (PARM);
76:     #_HDGS = 2;
77:     HEADING(1) = '         ACCIDENT FILE UPDATE -- EDIT PHASE';

78:     /* READ CITY TABLE */
79:     CNTY(0) = '00';
80:     OPEN FILE (TABLE) TITLE ('CITYTBL');
81:     DO I=1 TO 126;
82:     READ FILE (TABLE) SET (PTR_TBL);
83:        CNTY(I) = CITY.CNTY_#;
84:        END;
85:     CLOSE FILE (TABLE);

86:     /* OPEN FILES */
87:     OPEN FILE (DATA) TITLE ('EDITIN'),
88:        FILE (EDIT) TITLE ('EDITOUT');
89:     END_FLAG = ' ';
90:     ON ENDFILE (DATA) BEGIN;
91:        END_FLAG = 'X';
92:        GOTO TEST_CD;
93:        END;
94:     ERR_CNTR, ACC_CNTR = 0;
95:     #_ACC = 0;
96:     ACC_NOS = ' ';


97: /***** EXECUTION LOOP *****/

98: READ_DATA:
99:     IF END_FLAG¬=' ' THEN GOTO DONE;
```

```
/* ACCIDENT FILE EDIT PHASE */

100:    READ FILE (DATA) SET (PTR_DATA);

101: A_CARD:
102:    IF END_FLAG¬=' ' THEN GOTO DONE;

103:    /* FIRST CARD MUST BE 'A' OR 'E' CARD */
104:    IF CODE¬='A' & CODE¬='E' THEN GOTO A_ERROR;
105:    S_A = CARD;

106:    /* INIT VAR */
107:    Z1 = 2;
108:    ERR_FLAG, COR_FLAG = ' ';
109:    #_C,#_D,#_I = 0;
110:    S_C, S_D = ' ';
111:    S_I = ' ';

112: B_CARD:        /* 'B' CARD IS OPTIONAL */
113:    B_FLAG = ' ';
114:    READ FILE (DATA) SET (PTR_DATA);
115:    IF CODE='B' THEN DO;
116:       B_FLAG = 'X';
117:       S_B = CARD;
118:       READ FILE (DATA) SET (PTR_DATA);
119:       END;

120: C_CARD:
121:    IF (A.CODE='A' & CODE¬='C') | (A.CODE='E' & CODE¬='G') |
122:       SUBSTR(CARD,3,12)¬=A.KEY THEN GOTO I_CARD;
123:    #_C = #_C + 1;
124:    S_C(#_C) = CARD;
125:    READ FILE (DATA) SET (PTR_DATA);

126: D_CARD:
127:    IF (A.CODE='A' & CODE¬='D') | (A.CODE='E' & CODE¬='H') |
128:       SUBSTR(CARD,3,12)¬=A.KEY THEN GOTO I_CARD;
129:    #_D = #_D + 1;
130:    S_D(#_D) = CARD;
131:    READ FILE (DATA) SET (PTR_DATA);
132:    GOTO C_CARD;

133: I_CARD:
134:    IF CODE='I' THEN DO;
135:       #_I = #_I + 1;
136:       S_I(#_I) = CARD;
137:       READ FILE (DATA) SET (PTR_DATA);
138:       GOTO C_CARD;
139:       END;

140: TEST:

141:    /* STORE ACCIDENT NUMBER */
142:    ACC_#, SAVE_ACC_# = SUBSTR(S_A,3,12);
143:    IF #_ACC=0 THEN DO;  /* FIRST ACCIDENT CANNOT BE DUPLICATE */
144:       #_ACC = 1;
145:       ACC_NOS(1) = ACC_#;
146:       GOTO TEST_SEQUENCE;
147:       END;
148:    DUP_ACC = ' ';
```

```
/* ACCIDENT FILE EDIT PHASE */

149: SEARCH:

150:     /* ACCIDENT NUMBERS ARE STORED IN ASCENDING ORDER IN ARRAY.
151:        SEARCH FOR DUPLICATE NUMBER */
152:     DO I=1 TO #_ACC WHILE (ACC_#>ACC_NOS(I));
153:        END;

154:     /* TEST FOR DUPLICATE ACCIDENT NUMBER.  ASSIGN NEW NUMBER IF
155:        NECESSARY */
156:     IF ACC_#=ACC_NOS(I) THEN DO;
157:        DUP_ACC = 'X';
158:        IF ACC_#_SEQ=90 THEN GOTO TOO_MANY;
159:        IF ACC_#_SEQ>90
160:           THEN ACC_#_SEQ = ACC_#_SEQ - 1;
161:           ELSE ACC_#_SEQ = 99;
162:        GOTO SEARCH;
163:        END;

164:     /* STORE ACCIDENT NUMBER IN TABLE */
165:     IF I<=#_ACC THEN DO J=#_ACC TO I BY -1;
166:        ACC_NOS(J+1) = ACC_NOS(J);
167:        END;
168:     ACC_NOS(I) = ACC_#;
169:     #_ACC = #_ACC + 1;

170:     /* PRINT ERROR MESSAGE FOR DUPLICATE ACCIDENT NUMBERS */
171:     IF DUP_ACC='X' THEN DO;
172:        PRINTER =
173:           SAVE_ACC_# || ACC_# || (2)' ' ||
174:           SUBSTR(S_A,15,2) || '/' || SUBSTR(S_A,17,2) || '/' ||
175:           SUBSTR(S_A,19,2) || (2)' ' || SUBSTR(S_A,21,2) || ':' ||
176:           SUBSTR(S_A,23,2) || (2)' ' ||
177:           SUBSTR(S_A,25,3) || (2)' ' || SUBSTR(S_A,28,2) || (2)' ' ||
178:           SUBSTR(S_A,30,12) || '     *** DUPLICATE ADCIDENT NUMBER';
179:        CALL PRINTX (2);
180:        SUBSTR(S_A,3,12),SUBSTR(S_B,3,12) = ACC_#;
181:        DO I=1 TO #_C;
182:           SUBSTR(S_C(I),3,12), SUBSTR(S_D(I),3,12) = ACC_#;
183:           END;
184:        IF #_I¬=0 THEN DO I=1 TO #_I;
185:           SUBSTR(S_I(I),3,12) = ACC_#;
186:           END;
187:        ERR_CNTR = ERR_CNTR + 1;
188:        END;

189: TEST_SEQUENCE:

190:     /* IF NO CORRESPONDING 'D' CARD IS READ FOR A 'C' CARD, OR IF
191:        UNPROCESSED CARDS FOR AN ACCIDENT REMAIN, A CARD SEQUENCE
192:        ERROR EXISTS */
193:     IF #_C¬=#_D |
194:       (END_FLAG=' '&SUBSTR(CARD,3,12)=SAVE_ACC_#&CODE¬='A'&CODE¬='E'
195:        THEN DO;
196:        Z1 = 2;
197:        CALL SEQ_ERR(S_A);
198:        Z1 = 1;
199:        IF B_FLAG='X' THEN CALL SEQ_ERR(S_B);
```

```
/* ACCIDENT FILE EDIT PHASE */

200:        DO I=1 TO #_C;
201:           CALL SEQ_ERR(S_C(I));
202:           IF I<=#_D THEN CALL SEQ_ERR(S_D(I));
203:           END;
204:        IF #_I¬=0 THEN DO I=1 TO #_I;
205:           CALL SEQ_ERR(S_I(I));
206:           END;
207:        DO WHILE (SUBSTR(CARD,3,12)=SAVE_ACC_# & END_FLAG=' ');
208:           SUBSTR(CARD,3,12) = ACC_#;
209:           CALL SEQ_ERR(CARD);
210:           READ FILE (DATA) SET (PTR_DATA);
211:           END;
212:        ERR_CNTR = ERR_CNTR + 1;
213:        GOTO A_CARD;
214:        END;


215: /***** TEST 'A' CARD ****/

216:     /* TEST FOR NON-NUMERIC CHARS IN NUMERIC FIELDS */
217:     PTR = ADDR(S_A);
218:     DO I=3 TO 4, 6 TO 10, 12 TO 29, 42 TO 69, 71 TO 75;
219:        IF CHR(I)<'0' | CHR(I)>'9' THEN DO;
220:           PRINTER = S_A || '  NON-NUMERIC CHARACTER IN COLUMN' || I;
221:           CALL PRINTXA (Z1,2);
222:           PRINTER = SUBSTR(BLANKS,1,I-1) || '$';
223:           CALL PRINTX (1);
224:           ERR_FLAG = 'X';
225:           GOTO PRINT_ERROR;
226:           END;
227:        END;

228:     /* TEST FOR INVALID DATE */
229:     IF SUBSTR(A.KEY,1,2)¬=A.YEAR THEN DO;
230:        PRINTER = S_A || '  YEAR (19-20) IN ERROR';
231:        CALL PRINTXA (Z1,2);
232:        PRINTER = (18)' ' || '$$';
233:        CALL PRINTX (1);
234:        Z2 = '1';
235:        A.YEAR = SUBSTR(A.KEY,1,2);
236:        COR_FLAG = 'X';
237:        END;

238:     /* TEST FOR INVALID CITY NUMBER */
239:     IF A.CITY_#>126 THEN DO;
240:        PRINTER = S_A || ' CITY (25-27) TOO LARGE';
241:        CALL PRINTXA (Z1,2);
242:        PRINTER = (24)' ' || '$$$';
243:        CALL PRINTX (1);
244:        Z2 = '1';
245:        ERR_FLAG = 'X';
246:        END;

247:     /* TEST FOR INVALID COUNTY NUMBER */
248:     IF A.CNTY_#>'56' | A.CNTY_#='00' THEN DO;
249:        PRINTER = S_A || '  COUNTY (28-29) TOO LARGE OR ZERO';
250:        CALL PRINTXA (Z1,2);
251:        PRINTER = (27)' ' || '$$';
```

```
/* ACCIDENT FILE EDIT PHASE */

252:        CALL PRINTX (1);
253:        Z2 = '1';
254:        ERR_FLAG = 'X';
255:        END;

256:     /* CROSS-CHECK LOCATION */
257:     IF A.SYS_CODE='T' THEN A.SYS_CODE = 'M';
258:     IF A.SYS_CODE='U' THEN A.SYS_CODE = 'R';
259:     IF A.SYS_CODE='I'
260:        THEN IF A.RT_#='095' | A.RT_#='115' | A.RT_#='315'
261:           THEN A.RT_# = '015';
262:     IF SUBSTR(A.MILEPOST,4,1)='+' & A.CITY_#¬=0 THEN DO;
263:        PRINTER = S_A || '  CITY NUMBER (25-27) IN RURAL ACCIDENT';
264:        CALL PRINTXA (Z1,2);
265:        PRINTER = (24)' ' || '$$$';
266:        CALL PRINTX (1);
267:        Z2 = '1';
268:        A.CITY_# = 0;
269:        COR_FLAG = 'X';
270:        END;
271:     IF A.TRAF='2' | A.TRAF='5' THEN A.TRAF = '3';

272:     /* CROSS-CHECK CITY AND COUNTY NUMBER */
273:     IF A.CITY_#¬=0 & A.CNTY_#¬=CNTY(A.CITY_#) THEN DO;
274:        PRINTER = S_A || '  COUNTY (28-29) AND CITY (25-27) DISAGREE';
275:        CALL PRINTXA (Z1,2);
276:        PRINTER = (24)' ' || '$$$$$';
277:        CALL PRINTX (1);
278:        Z2 = '1';
279:        A.CNTY_# = CNTY(A.CITY_#);
280:        COR_FLAG = 'X';
281:        END;


282: /***** TEST "B" CARD *****/

283:     IF B_FLAG='X' THEN DO;

284:        /* TEST NUMERIC FIELDS */
285:        PTR = ADDR(S_B);
286:        DO I=15 TO 34;
287:           IF CHR(I)<'0' | CHR(I)>'9' THEN DO;
288:              PRINTER = S_B || '  NON-NUMERIC CHARACTER IN COLUMN' ||
289:              CALL PRINTXA (Z1,2);
290:              PRINTER = SUBSTR(BLANKS,1,I-1) || '$';
291:              CALL PRINTX (1);
292:              Z2 = '1';
293:              ERR_FLAG = 'X';
294:           GOTO PRINT_ERROR;
295:              END;
296:           END;
297:        END;

298:     /* TEST NUMERIC FIELDS ON "C" CARDS */
299: TEST_CD:
300:     DO I=1 TO #_C;
301:        PTR = ADDR(S_C(I));
302:        DO J=56 TO 61;
```

```
/* ACCIDENT FILE EDIT PHASE */

303:          IF CHR(J)<'0' | CHR(J)>'9' THEN DO;
304:             PRINTER = S_C(I) ||
305:                '  NON-NUMERIC CHARACTER IN COLUMN' || J;
306:             CALL PRINTXA (Z1,2);
307:             PRINTER = SUBSTR(BLANKS,1,J-1) || '$';
308:             CALL PRINTX (1);
309:             Z2 = '1';
310:             ERR_FLAG = 'X';
311:             GOTO PRINT_ERROR;
312:             END;
313:          END;
314:       END;

315:    /* TEST NUMERIC FIELDS ON "D" CARDS */
316:    DO I=1 TO #_D;
317:       PTR = ADDR(S_D(I));
318:       DO J=15 TO 22, 24 TO 27, 29 TO 32, 34 TO 37,
319:          39 TO 42, 44 TO 47, 49 TO 62, 80;
320:          IF CHR(J)<'0' | CHR(J)>'9' THEN DO;
321:             PRINTER = S_D(I) ||
322:                '  NON-NUMERIC CHARACTER IN COLUMN' || J;
323:             CALL PRINTXA (Z1,2);
324:             PRINTER = SUBSTR(BLANKS,1,J-1) || '$';
325:             CALL PRINTX (1);
326:             Z2 = '1';
327:             ERR_FLAG = 'X';
328:          GOTO PRINT_ERROR;
329:             END;
330:          END;
331:       END;

332:    /* TEST NUMERIC FIELDS ON "I" CARDS */
333:    IF #_I¬=0 THEN DO I=1 TO #_I;
334:       PTR = ADDR(S_I(I));
335:       DO J=20 TO 22,24 TO 27,29 TO 32,34 TO 37,39 TO 42,44 TO 47,
336:          49 TO 51;
337:          IF CHR(J)<'0' | CHR(J)>'9' THEN DO;
338:             PRINTER = S_I(I) ||
339:                '  NON-NUMERIC CHARACTER IN COLUMN' || J;
340:             CALL PRINTXA (Z1,2);
341:             PRINTER = SUBSTR(BLANKS,1,J-1) || '$';
342:             CALL PRINTX (1);
343:             Z2 = '1';
344:             ERR_FLAG = 'X';
345:             GOTO PRINT_ERROR;
346:             END;
347:          END;
348:       END;

349:    /* CALCULATE NUMBER OF VEHS, PEDS, INJS, AND FATS */
350:    #_VEH, #_PED, #_FAT, #_INJ = 0;
351:    INJ_SEV, DAM_SEV = '0';
352:    DO I = 1 TO #_C;
353:       IF SUBSTR(S_D(I),50,2)¬='00' THEN DO;
354:          IF SUBSTR(S_D(I),54,2)¬='00' THEN DO;
355:             PRINTER = S_D(I) ||
356:                '  BOTH VEH (50-51) AND PED (54-55) NUMBER CODED';
357:             CALL PRINTXA (Z1,2);
```

```
/* ACCIDENT FILE EDIT PHASE */

358:              PRINTER = (49)' ' || '$$  $$';
359:              CALL PRINTX (1);
360:              Z2 = '1';
361:              COR_FLAG = 'X';
362:              END;
363:           #_VEH = #_VEH + 1;
364:           END;
365:        ELSE DO;
366:           IF SUBSTR(S_D(I),54,2)='00' THEN DO;
367:              PRINTER = S_D(I) ||
368:                 '  NEITHER VEH (50-51) NOR PED (54-55) NUMBER CODED'
369:              CALL PRINTXA (Z1,2);
370:              PRINTER = (49)' ' || '$$  $$';
371:              CALL PRINTX (1);
372:              Z2 = '1';
373:              COR_FLAG = 'X';
374:              END;
375:           #_PED = #_PED + 1;
376:           END;
377:        PTR = ADDR(S_D(I));
378:        DO J=24 TO 49 BY 5;
379:           IF CHR(J)='1' THEN #_FAT = #_FAT + 1;
380:           IF CHR(J)>'1' THEN #_INJ = #_INJ + 1;
381:           IF CHR(J)>'0' & (CHR(J)<INJ_SEV | INJ_SEV='0')
382:              THEN INJ_SEV = CHR(J);
383:           END;
384:        IF CHR(80)>'0' & (CHR(80)<DAM_SEV | DAM_SEV='0')
385:           THEN DAM_SEV = CHR(80);
386:        END;
387:     IF #_I¬=0 THEN DO I=1 TO #_I;
388:        PTR = ADDR(S_I(I));
389:        DO J=24 TO 49 BY 5;
390:           IF CHR(J)='1' THEN #_FAT = #_FAT + 1;
391:           IF CHR(J)>'1' THEN #_INJ = #_INJ + 1;
392:           IF CHR(J)>'0' & (CHR(J)<INJ_SEV | INJ_SEV='0')
393:              THEN INJ_SEV = CHR(J);
394:           END;
395:        END;

396:     /* CONVERT #_VEH AND #_PED TO DECIMAL */
397:     DEC_#_VEH = A.#_VEH;
398:     DEC_#_PED = A.#_PED;

399:     /* CHECK NUMBER OF VEHICLES */
400:     IF #_VEH¬=DEC_#_VEH THEN DO;
401:        PRINTER = S_A || '  NUMBER OF VEHICLES (51-52) IN ERROR';
402:        CALL PRINTXA (Z1,2);
403:        PRINTER = (50)' ' || '$$';
404:        CALL PRINTX (1);
405:        Z2 = '1';
406:        COR_FLAG = 'X';
407:        A.#_VEH = #_VEH;
408:        END;

409:     /* CHECK NUMBER OF PEDESTRIANS */
410:     IF #_PED¬=DEC_#_PED THEN DO;
411:        PRINTER = S_A || '  NUMBER OF PEDESTRIANS (53-54) IN ERROR';
412:        CALL PRINTXA (Z1,2);
```

```
/* ACCIDENT FILE EDIT PHASE */

413:        PRINTER = (52)' ' || '$$';
414:        CALL PRINTX (1);
415:        Z2 = '1';
416:        COR_FLAG = 'X';
417:        A.#_PED = #_PED;
418:        END;

419:     /* CHECK NUMBER OF FATALITIES */
420:     IF #_FAT¬=A.#_FAT THEN DO;
421:        PRINTER = S_A || '  NUMBER OF FATALITIES (55-56) IN ERROR';
422:        CALL PRINTXA (Z1,2);
423:        PRINTER = (54)' ' || '$$';
424:        CALL PRINTX (1);
425:        Z2 = '1';
426:        COR_FLAG = 'X';
427:        A.#_FAT = #_FAT;
428:        END;

429:     /* CHECK NUMBER OF INJURIES */
430:     IF #_INJ¬=A.#_INJ THEN DO;
431:        PRINTER = S_A || '  NUMBER OF INJURIES (57-58) IN ERROR';
432:        CALL PRINTXA (Z1,2);
433:        PRINTER = (56)' ' || '$$';
434:        CALL PRINTX (1);
435:        Z2 = '1';
436:        COR_FLAG = 'X';
437:        A.#_INJ = #_INJ;
438:        END;

439:     /* CHECK DAMAGE SEVERITY */
440:     IF DAM_SEV¬=A.DAM_SEV THEN DO;
441:        PRINTER = S_A || '  DAMAGE SEVERITY (47) IN ERROR';
442:        CALL PRINTXA (Z1,2);
443:        PRINTER = (46)' ' || '$';
444:        CALL PRINTX (1);
445:        Z2 = '1';
446:        COR_FLAG = 'X';
447:        A.DAM_SEV = DAM_SEV;
448:        END;

449:     /* CHECK INJURY SEVERITY */
450:     IF INJ_SEV¬=A.INJ_SEV THEN DO;
451:        PRINTER = S_A || '  INJURY SEVERITY (46) IN ERROR';
452:        CALL PRINTXA (Z1,2);
453:        PRINTER = (45)' ' || '$';
454:        CALL PRINTX (1);
455:        Z2 = '1';
456:        COR_FLAG = 'X';
457:        A.INJ_SEV = INJ_SEV;
458:        END;

459:     /* TEST FOR ZERO VEHICLES */
460:     IF #_VEH=0 THEN DO;
461:        PRINTER = S_A || '  NUMBER OF VEHICLES (51-52) IS ZERO';
462:        CALL PRINTXA (Z1,2);
463:        PRINTER = (50)' ' || '$$';
464:        CALL PRINTX (1);
465:        Z2 = '1';
```

```
/* ACCIDENT FILE EDIT PHASE */

466:        ERR_FLAG = 'X';
467:        END;

468:     /* WRITE ACCIDENTS WITHOUT ERRORS */
469:     IF ERR_FLAG=' ' & COR_FLAG=' ' THEN DO;
470:        OUT = S_A;
471:        WRITE FILE (EDIT) FROM(OUT);
472:        OUT = S_B;
473:        IF B_FLAG='X' THEN WRITE FILE (EDIT) FROM (OUT);
474:        DO I=1 TO #_VEH+#_PED;
475:           OUT = S_C(I);
476:           WRITE FILE (EDIT) FROM (OUT);
477:           OUT = S_D(I);
478:           WRITE FILE (EDIT) FROM (OUT);
479:           END;
480:        IF #_I¬=0 THEN DO I=1 TO #_I;
481:           OUT = S_I(I);
482:           WRITE FILE (EDIT) FROM (OUT);
483:           END;
484:        GOTO A_CARD;
485:        END;

486: PRINT_ERROR:

487:     /* WRITE ACCIDENTS CONTAINING ERRORS */
488:     PRINTER = 'ACCIDENT-NUMBER=' || SUBSTR(S_A,3,12) ||
489:        ', DATE=' || SUBSTR(S_A,15,2) || '/' || SUBSTR(S_A,17,2) ||
490:        '/' || SUBSTR(S_A,19,2) ||
491:        ', TIME=' || SUBSTR(S_A,21,2) || ':' || SUBSTR(S_A,23,2);
492:     CALL PRINTXA (Z1,2);
493:     PRINTER = 'COUNTY=' || SUBSTR(S_A,28,2) ||
494:        ', CITY=' || SUBSTR(S_A,25,3) ||
495:        ', MILEPOST=' || SUBSTR(S_A,30,12);
496:     CALL PRINTX (1);
497:     OUT = S_A || ERR_FLAG;
498:     WRITE FILE (EDIT) FROM (OUT);
499:     PRINTER = S_A || '  ***';
500:     CALL PRINTX (1);
501:     IF B_FLAG¬=' ' THEN DO;
502:        OUT = S_B || ERR_FLAG;
503:        WRITE FILE (EDIT) FROM (OUT);
504:        SUBSTR(PRINTER,1,80) = S_B;
505:        CALL PRINTX (1);
506:        END;
507:     IF #_VEH+#_PED¬=0 THEN DO I=1 TO #_VEH+#_PED;
508:        OUT = S_C(I) || ERR_FLAG;
509:        WRITE FILE (EDIT) FROM (OUT);
510:        SUBSTR(PRINTER,1,80) = OUT;
511:        CALL PRINTX (1);
512:        OUT = S_D(I);
513:        WRITE FILE (EDIT) FROM (OUT);
514:        SUBSTR(PRINTER,1,80) = OUT;
515:        CALL PRINTX (1);
516:        END;
517:     IF #_I¬=0 THEN DO I=1 TO #_I;
518:        OUT = S_I(I);
519:        WRITE FILE (EDIT) FROM (OUT);
520:        SUBSTR(PRINTER,1,80) = OUT;
```

```
/* ACCIDENT FILE EDIT PHASE */

521:        CALL PRINTX (1);
522:        END;
523:     ERR_CNTR = ERR_CNTR + 1;
524:     GOTO A_CARD;

525:     /* ERR MSG:  FIRST CARD NOT "A" OR "E" CARD */

526: A_ERROR:
527:     PRINTER = CARD || '  *** "A" CARD MISSING';
528:     CALL PRINTX (2);
529:     OUT = CARD || 'X';
530:     WRITE FILE (EDIT) FROM (OUT);
531:     ERR_CNTR = ERR_CNTR + 1;
532:     READ FILE (DATA) SET (PTR_DATA);
533:     DO WHILE (CODE¬='A' & CODE¬='E' & END_FLAG=' ');
534:        SUBSTR(PRINTER,1,80) = CARD;
535:        CALL PRINTX (1);
536:        OUT = CARD || 'X';
537:        WRITE FILE (EDIT) FROM (OUT);
538:        READ FILE (DATA) SET (PTR_DATA);
539:        END;
540:     GOTO A_CARD;


541: /***** ERR MSG -- CANNOT FORM UNIQUE ACCIDENT NUMBER *****/

542: TOO_MANY:
543:     PRINTER = S_A || '   *** DUPLICATE ACCIDENT NUMBER';
544:     CALL PRINTX (2);
545:     GOTO A_CARD;


546: /***** SUBROUTINE TO PRINT SEQUENCE NUMBER ERROR MESSAGE *****/

547: SEQ_ERR:  PROCEDURE (C);

548: DECLARE C CHAR(80);

549:     SUBSTR(C,3,12) = ACC_#;
550:     PRINTER = C || '  *** SEQUENCE ERROR';
551:     CALL PRINTX (Z1);
552:     OUT = C || 'X';
553:     WRITE FILE (EDIT) FROM (OUT);
554:     END SEQ_ERR;


555: DONE:
556:     PRINTER = '   END OF DATA.';
557:     CALL PRINTX (3);
558:     PRINTER = '   TOTAL NUMBER OF ACCIDENTS:' || #_ACC;
559:     CALL PRINTX (1);
560:     PRINTER = '   NUMBER OF ACCIDENTS IN ERROR:' || ERR_CNTR;
561:     CALL PRINTX (1);
562:     CLOSE FILE (DATA), FILE (EDIT);
563:     CALL EXIT (PARM);

564: END EDITOR;
```

UPDATE-ERRORS --

Member Name	PZA	
Language	PL/I	
Subroutines	none	
Files	SYSPRINT	-- IBM and PZA messages
	CORIN	-- User-supplied update cards
	EDITIN	-- Output file
	EDITOUT	-- Error file being updated
Instruction	1 - 3	"PZA"

UPDATE-ERRORS is used for correcting accident data cards rejected by EDIT-DATA-CARDS. Functions are available allowing insertion, deletion, and revision. The program reads the user-supplied update cards and the edit output file sequentially, building an output file of updated data. It is thus imperative that the update cards be set up in the same order that the corresponding records appear in the error file. Records are identified by an accident key, the first 14 characters (card code, sequence number, and 12-character accident number). Each update card must have this key coded in the first 14 columns. When an update card is read, the program searches for a record with that key by reading sequentially through the error file, copying each record bypassed into the output file. After the card is read, the appropriate function is performed, and the next update card read. The updating functions are described in the publication Highway Information System Volume 1: User Information.

The PZA program listing follows:

```
/* :UPDATE-ERRORS */

 1: /* :UPDATE-ERRORS */

 2: UPDATE:  PROCEDURE OPTIONS (MAIN);

 3: /* EDITOUT FILE -- INPUT RECORDS */
 4: DECLARE
 5:    ERR CHAR(81) BASED (PTR_ERR),
 6:    ECHR(80) CHAR(1) BASED (PTR_ERR),
 7:    ERROR FILE RECORD INT;

 8: /* DATA FILE -- INPUT DATA */
 9: DECLARE
10:    CARD CHAR(80) BASED (PTR_IN),
11:    CHR(80) CHAR(1) BASED (PTR_IN),
12:    1  C BASED (PTR_IN),
13:       2  DUM1 CHAR(14),
14:       2  TYPE CHAR(1),
15:    DATA FILE RECORD INT;

16: /* OUTPUT RECORDS */
17: DECLARE
18:    C81 CHAR(81),
19:    OUT FILE RECORD INT OUTPUT ENV (F(3483,81));

20: /* OTHER VARIABLES */
21: DECLARE
22:    I_FLAG CHAR(1),
23:   (FLAG,READ_FLAG) CHAR(1);


24: /***** INITIALIZATION *****/

25:    OPEN FILE (DATA) TITLE ('CORIN');
26:    ON ENDFILE (DATA) GOTO END_DATA;
27:    READ FILE (DATA) SET (PTR_IN);
28:    FLAG = 'X';

29: RESTART:
30:    OPEN
31:       FILE (ERROR) TITLE ('EDITOUT'),
32:       FILE (OUT) TITLE ('EDITIN');
33:    ON ENDFILE (ERROR) GOTO END_ERROR;
34:    READ FILE (ERROR) SET (PTR_ERR);


35: /***** MAIN LOOP *****/

36: LOOP:

37:    /* FIND SPECIFIED RECORD */
38:    DO WHILE (SUBSTR(ERR,1,14)¬=SUBSTR(CARD,1,14));
39:       WRITE FILE (OUT) FROM (ERR);
40:       READ FILE (ERROR) SET (PTR_ERR);
41:       END;

42:    /* CHECK FOR UPDATE FUNCTION */
43:    READ_FLAG = ' ';
44:    IF C.TYPE='¬' THEN GOTO DELETE;
```

```
/* :UPDATE-ERRORS */

45:     IF C.TYPE='=' THEN GOTO NEW_KEY;
46:     IF C.TYPE='*' | C.TYPE='%' THEN GOTO INSERT;

47:     /* REWRITE FUNCTION */
48:     DO I=15 TO 80;
49:        IF CHR(I)¬=' '
50:           THEN IF CHR(I)='$'
51:              THEN ECHR(I) = ' ';
52:              ELSE ECHR(I) = CHR(I);
53:        END;
54:     WRITE FILE (OUT) FROM (ERR);
55:     READ_FLAG = 'X';
56:     GOTO READ_DATA;

57:     /* DELETE FUNCTION */
58: DELETE:
59:     READ_FLAG = 'X';
60:     GOTO READ_DATA;

61:     /* NEW-KEY FUNCTION */
62: NEW_KEY:
63:     SUBSTR(ERR,1,14) = SUBSTR(CARD,16,14);
64:     GOTO READ_DATA;

65:     /* INSERT FUNCTION */
66: INSERT:
67:     READ FILE (DATA) SET (PTR_IN);
68:     IF C.TYPE='%' THEN DO;
69:        C81 = CARD;
70:        WRITE FILE (OUT) FROM (C81);
71:        READ_FLAG = 'X' ;
72:        END;
73:     I_FLAG = ' ';
74:     IF CHR(1)='>' THEN DO;
75:        I_FLAG = '>';
76:        READ FILE (DATA) SET (PTR);
77:        END;
78: INSERT_LOOP:
79:     C81 = CARD;
80:     WRITE FILE (OUT) FROM (C81);
81:     IF I_FLAG=' ' THEN GOTO READ_DATA;
82:     READ FILE (DATA) SET (PTR_IN);
83:     IF CHR(1)='>' THEN GOTO READ_DATA;
84:     GOTO INSERT_LOOP;

85:     /* GET NEXT DATA CARD & REPEAT LOOP */
86: READ_DATA:
87:     READ FILE (DATA) SET (PTR_IN);
88:     IF READ_FLAG='X' THEN READ FILE (ERROR) SET (PTR_ERR);
89:     FLAG = ' ';
90:     GOTO LOOP;


91: /***** EOF ON ERROR WHILE PROCESSING DATA CARD *****/

92: END_ERROR:
93:     IF FLAG='X' THEN DO;
94:        PUT FILE (SYSPRINT) SKIP EDIT
```

```
/* :UPDATE-ERRORS */

 95:          (CARD,'  ****RECORD DOES NOT EXIST****') (A);
 96:       ON ENDFILE (DATA) GOTO DONE;
 97:       READ FILE (DATA) SET (PTR_IN);
 98:       ON ENDFILE (DATA) GOTO END_DATA;
 99:       FLAG = ' ';
100:       END;
101:    ELSE FLAG = 'X';
102:    CLOSE FILE (ERROR), FILE (OUT);
103:    OPEN
104:       FILE (OUT) INPUT TITLE ('EDITIN'),
105:       FILE (ERROR) OUTPUT TITLE ('EDITOUT');
106:    ON ENDFILE (OUT) GOTO EOFE2;

107: EOFE1:
108:    READ FILE (OUT) SET (PTR_ERR);
109:    WRITE FILE (ERROR) FROM (ERR);
110:    GOTO EOFE1;

111: EOFE2:
112:    CLOSE FILE (OUT), FILE (ERROR);
113:    GOTO RESTART;


114: /***** EOF ON DATA FILE *****/

115: END_DATA:
116:    IF READ_FLAG=' ' THEN WRITE FILE (OUT) FROM (ERR);
117:    ON ENDFILE (ERROR) GOTO DONE;

118: EOFD1:
119:    READ FILE (ERROR) SET (PTR_ERR);
120:    WRITE FILE (OUT) FROM (ERR);
121:    GOTO EOFD1;


122: /***** EXECUTION COMPLETE *****/

123: DONE:
124:    PUT FILE (SYSPRINT) SKIP(2) EDIT ('END OF DATA.') (A);
125:    CLOSE FILE (OUT), FILE (DATA), FILE (ERROR);

126: END UPDATE;
```

STORE-DATA-CARDS --

Member Name	PXA
Language	PL/I
Subroutines	none
Files	SYSPRINT -- IBM and PXA messages EDITOUT -- Accident data cards TAPEOUT -- Output file
Instruction	1 - 3 "PXA"

STORE-DATA-CARDS copies the edit program output file into a sequential tape, disk, or card file for backup purposes. Records in error are flagged by EDIT-DATA-CARDS by an "X" in column 81. These records are not copied.

The PXA program listing follows:

```
/* :STORE-DATA-CARDS */

 1: /* :STORE-DATA-CARDS */

 2: STORE:  PROCEDURE OPTIONS (MAIN);

 3: /* DATA INPUT/OUTPUT */
 4: DECLARE
 5:    1  S BASED (PTR),
 6:       2  CARD CHAR(80),
 7:       2  X CHAR(1),
 8:    CRD CHAR(80) BASED (PTR),
 9:    EDITOUT FILE INT RECORD,
10:    TAPEOUT FILE INT RECORD OUTPUT ENV (F(3600,80));


11: /***** INITIALIZATION *****/

12:    OPEN
13:       FILE (EDITOUT),
14:       FILE (TAPEOUT);
15:    ON ENDFILE (EDITOUT) GOTO DONE;
16:    I = 0;


17: /***** MAIN LOOP *****/

18: LOOP:
19:    READ FILE (EDITOUT) SET (PTR);
20:    IF S.X='X' THEN GOTO LOOP;
21:    WRITE FILE (TAPEOUT) FROM (CRD);
22:    I = I + 1;
23:    GOTO LOOP;


24: /***** EXECUTION END *****/

25: DONE:
26:    CLOSE FILE (EDITOUT), FILE (TAPEOUT);
27:    PUT FILE (SYSPRINT) SKIP EDIT
28:       ('STORE-DATA-CARDS SUCCESSFULLY TERMINATED') (A);
29:    PUT FILE (SYSPRINT) SKIP(2) EDIT
30:       ('NUMBER OF DATA CARDS COPIED:',I) (A);

31: END STORE;
```

LOAD-ACCIDENT-DATA --

Member Name	PWA
Language	PL/I
Subroutines	CONVACC
Files	SYSPRINT -- IBM and PWA messages EDITOUT -- Data cards ACIDENTM -- Detail file ACCVEHM -- Vehicle file MEMOS -- Memos file
Instruction	1 - 3 "PWA"

LOAD-ACCIDENT-DATA loads a file of sorted accident cards into two sequential files: a detail file (with information from the "A" and "B" cards) and a vehicle file (with information from the "C," "D," and "I" cards). Subroutine CONVACC performs the conversion from data cards to detail and vehicle records. LOAD-ACCIDENT-DATA simply reads the cards, invokes CONVACC to perform the conversion, and writes the records into output files ACIDENTM and ACCVEHM.

The PWA program listing follows:

```
/* :LOAD-ACCIDENT-DATA */

 1: /* :LOAD-ACCIDENT-DATA */

 2: LOAD:  PROCEDURE OPTIONS (MAIN);

 3: /* INPUT-OUTPUT VARIABLES */
 4: DECLARE
 5:    CODE CHAR(1) BASED (PTR_DATA),
 6:    CARD CHAR(81) BASED (PTR_DATA),
 7:    1  CARD_STR BASED (PTR_DATA),
 8:       2  DUM1 CHAR(80),
 9:       2  X CHAR(1),
10:   (S_A,S_B,S_C(50),S_D(50),S_I(20)) CHAR(80) STATIC,
11:    1  A DEF S_A,
12:       2  DUM1 CHAR(2),
13:       2  KEY CHAR(12),
14:    DETAIL CHAR(96) STATIC,
15:   (VEHICLE,VEHS(70)) CHAR(136) STATIC,
16:    DATA FILE INT RECORD,
17:   (ACIDENT,ACCVEH) FILE INT RECORD OUTPUT;

18: /* MEMOS FILE */
19: DECLARE
20:    M CHAR(82),
21:    MEMOS FILE INT RECORD OUTPUT ENV (F(3444,82));

22: /* OTHER VARIABLES */
23: DECLARE
24:    END_FLAG CHAR(1),
25:    VEH_# PIC'99',
26:    CNTR DEC FIXED (5,0),
27:   (#_VEH,#_PED,VEH_IND(50),PED_IND(50),#_C,#_D) DEC FIXED (3,0),
28:    #_I DEC FIXED (3,0),
29:   (I,J) DEC FIXED (7,0);


30: /***** INITIALIZATION *****/

31:    OPEN
32:       FILE (DATA) TITLE ('EDITOUT'),
33:       FILE (MEMOS),
34:       FILE (ACIDENT) TITLE ('ACIDENTM'),
35:       FILE (ACCVEH) TITLE ('ACCVEHM');
36:    END_FLAG = ' ';
37:    CNTR = 0;
38:    ON ENDFILE (DATA) BEGIN;
39:       END_FLAG = 'X';
40:       GOTO GOT_THEM;
41:       END;


42: /***** MAIN LOOP *****/

43:    /*** READ AN ACCIDENT ***/
44: READ_DATA:
45:    IF END_FLAG¬=' ' THEN GOTO DONE;
46:    READ FILE (DATA) SET (PTR_DATA);

47: A_CARD:
```

```
/* :LOAD-ACCIDENT-DATA */

 48:     IF END_FLAG¬=' ' THEN GOTO DONE;
 49:     IF X='X' THEN GOTO READ_DATA;
 50:     IF CODE¬='A' & CODE¬='E' THEN GOTO A_ERROR;
 51:     S_A = CARD;

 52: B_CARD:
 53:     READ FILE (DATA) SET (PTR_DATA);
 54:     IF CODE='B' THEN DO;
 55:        S_B = CARD;
 56:        READ FILE (DATA) SET (PTR_DATA);
 57:        END;
 58:     ELSE S_B = 'B0' || A.KEY || (20)'0';
 59:     #_C, #_D, #_I = 0;

 60: C_CARD:
 61:     IF CODE¬='C' & CODE¬='G' THEN GOTO I_CARD;
 62:     #_C = #_C + 1;
 63:     S_C(#_C) = CARD;
 64:     READ FILE (DATA) SET (PTR_DATA);

 65: D_CARD:
 66:     IF CODE¬='D' & CODE¬='H' THEN GOTO I_CARD;
 67:     #_D = #_D + 1;
 68:     S_D(#_D) = CARD;
 69:     READ FILE (DATA) SET (PTR_DATA);
 70:     GOTO C_CARD;

 71: I_CARD:
 72:     IF CODE='I' THEN DO;
 73:        #_I = #_I + 1;
 74:        S_I(#_I) = CARD;
 75:        READ FILE (DATA) SET (PTR_DATA);
 76:        GOTO C_CARD;
 77:        END;

 78:     /* TEST FOR SEQUENCE ERROR */
 79: GOT_THEM:
 80:     IF #_C¬=#_D | (END_FLAG=' ' & SUBSTR(CARD,3,12)=SUBSTR(S_A,3,12))
 81:        THEN DO;
 82:        PUT FILE (SYSPRINT) SKIP EDIT
 83:           ('*** SEQUENCE ERROR IN ACCIDENT ',S_A) (A);
 84:        GOTO A_CARD;
 85:        END;

 86:     /* FIND VEHICLE AND PEDESTRIAN RECORDS */
 87:     VEH_IND, PED_IND = 0;
 88:     #_VEH, #_PED = 0;
 89:     DO I=1 TO #_C;
 90:        IF SUBSTR(S_D(I),50,2)¬='00' THEN DO;
 91:           #_VEH = #_VEH + 1;
 92:           VEH_IND(#_VEH) = I;
 93:           END;
 94:        ELSE DO;
 95:           #_PED = #_PED + 1;
 96:           PED_IND(#_PED) = I;
 97:           END;
 98:        END;
```

```
/* :LOAD-ACCIDENT-DATA */

 99:     /* PERFORM CONVERSIONS */
100:     CALL CONVABR (S_A, S_B, DETAIL, CONV_ERROR);
101:     DO I=1 TO #_VEH;
102:        CALL CONVCDR (S_C(VEH_IND(I)),S_D(VEH_IND(I)),VEHS(I),
103:           DETAIL,CONV_ERROR);
104:        END;
105:     IF #_PED¬=0 THEN DO I=1 TO #_PED;
106:        CALL CONVCDR (S_C(PED_IND(I)),S_D(PED_IND(I)),VEHS(I+#_VEH),
107:           DETAIL,CONV_ERROR);
108:        END;
109:     IF #_I¬=0 THEN DO;
110:        CALL CONVCDR ((80)' ',S_I(I),VEHS(I+#_VEH+#_PED),DETAIL,
111:           CONV_ERROR);
112:        END;

113:     /* WRITE THE RECORDS */
114:     WRITE FILE (ACIDENT) FROM (DETAIL);
115:     DO I=1 TO #_C+#_I;
116:        VEHICLE = VEHS(I);
117:        IF SUBSTR(VEHICLE,14,1)='A' THEN VEH_# = I;
118:           ELSE IF SUBSTR(VEHICLE,14,1)='B' THEN VEH_# = I-#_VEH;
119:           ELSE VEH_# = I - #_C;
120:        SUBSTR(VEHICLE,15,2) = VEH_#;
121:        WRITE FILE (ACCVEH) FROM (VEHICLE);
122:        END;
123:     CNTR = CNTR + 1;
124:     IF SUBSTR(DETAIL,94,1)¬='X' THEN GOTO A_CARD;

125:     /* WRITE THE MEMO RECORDS */
126:     M = SUBSTR(S_A,3,18) || SUBSTR(S_A,28,14) || SUBSTR(S_A,51,2);
127:     IF SUBSTR(S_A,55,2)¬='00' THEN SUBSTR(M,35,1) = '2';
128:        ELSE SUBSTR(M,35,1) = '3';
129:     DO I=1 TO #_VEH;
130:        IF SUBSTR(S_C(VEH_IND(I)),15,22)¬=' ' THEN DO;
131:           SUBSTR(M,36) = SUBSTR(S_C(VEH_IND(I)),15,41) ||
132:              SUBSTR(S_C(VEH_IND(I)),63,6);
133:           WRITE FILE (MEMOS) FROM (M);
134:           END;
135:        END;
136:     GOTO A_CARD;

137: A_ERROR:
138:     PUT FILE (SYSPRINT) SKIP EDIT
139:        ('*** "A" CARD EXPECTED ',S_A) (A);
140:     GOTO READ_DATA;

141: CONV_ERROR:
142:     PUT FILE (SYSPRINT) SKIP EDIT
143:        ('*** CONVERSION ***   ', S_A) (A);
144:     GOTO A_CARD;


145: DONE:
146:     PUT FILE (SYSPRINT) SKIP(3) EDIT
147:        ('   END OF DATA.') (A);
148:     PUT FILE (SYSPRINT) SKIP(2) EDIT
149:        ('NUMBER OF ACCIDENTS LOADED:',CNTR) (A);
```

```
/* :LOAD-ACCIDENT-DATA */
150:    CLOSE FILE (ACIDENT), FILE(ACCVEH), FILE(DATA), FILE(MEMOS);
151: END LOAD;
```

MERGE-ACCIDENT-FILES --

Member Name	PVA	
Language	PL/I	
Subroutines	none	
Files	SYSPRINT	-- IBM and PVA messages
	ACIDENTM	-- Detail edit file
	ACCVEHM	-- Vehicle edit file
	ACIDENT	-- Full detail file
	ACCVEH	-- Full vehicle file
	TEMPFILE	-- Scratch file
Instruction	1 - 3	"PVA"

MERGE-ACCIDENT-FILES merges the edit files created by LOAD-ACCIDENT-DATA with the full detail and vehicle files. Merging proceeds accident-by-accident. Should a duplicate accident number occur in the full file and the edit file, the accident in the edit file replaces that in the full file. The program first merges the detail files. The edit file is merged with the full file, and the resultant file placed in TEMPFILE. After merging, the scratch file is copied back into ACIDENT. The process is repeated for the vehicle files, TEMPFILE again used for a scratch file.

The PVA program listing follows:

```
/* :MERGE-ACCIDENT-FILES */

  1: /* :MERGE-ACCIDENT-FILES */

  2: MERGE:  PROCEDURE OPTIONS (MAIN);

  3: /* ACCIDENT FILES */
  4: DECLARE
  5:    (D1,D2) CHAR(96),
  6:    (V1,V2) CHAR(136),
  7:     KD1 CHAR(12) DEF D1 POS(2),
  8:     KD2 CHAR(12) DEF D2 POS(2),
  9:     KV1 CHAR(12) DEF V1 POS(2),
 10:     KV2 CHAR(12) DEF V2 POS(2),
 11:     KV2A CHAR(15) DEF V2 POS(2),
 12:     ACC1 FILE INT RECORD,
 13:     ACC2 FILE INT KEYED RECORD ENV (INDEXED),
 14:     TEMPD FILE INT RECORD ENV (F(3456,96)),
 15:     TEMPV FILE INT RECORD ENV (F(3400,136));
 16:     PUT FILE (SYSPRINT) SKIP EDIT ('MERGE-ACCIDENT-FILES ROUTINE') (

 17: /*** DETAIL FILE ***/

 18:     OPEN
 19:        FILE (ACC1) TITLE ('ACIDENTM'),
 20:        FILE (ACC2) TITLE ('ACIDENT'),
 21:        FILE (TEMPD) OUTPUT TITLE ('TEMPFILE');
 22:     ON ENDFILE (ACC1) GOTO EOFD1;
 23:     ON ENDFILE (ACC2) GOTO EOFD2;
 24:     READ FILE (ACC2) INTO (D2);

 25: LOOPD:
 26:     READ FILE (ACC1) INTO (D1);
 27:     DO WHILE (KD2<KD1);
 28:        WRITE FILE (TEMPD) FROM (D2);
 29:        READ FILE (ACC2) INTO (D2);
 30:        END;
 31:     IF KD2=KD1 THEN READ FILE (ACC2) INTO (D2);
 32:     WRITE FILE (TEMPD) FROM (D1);
 33:     GOTO LOOPD;

 34: EOFD1:
 35:     ON ENDFILE (ACC2) GOTO CLOSED;

 36: LOOPD1:
 37:     WRITE FILE (TEMPD) FROM (D2);
 38:     READ FILE (ACC2) INTO (D2);
 39:     GOTO LOOPD1;

 40: EOFD2:
 41:     ON ENDFILE (ACC1) GOTO CLOSED;

 42: LOOPD2:
 43:     WRITE FILE (TEMPD) FROM (D1);
 44:     READ FILE (ACC1) INTO (D1);
 45:     GOTO LOOPD2;

 46: CLOSED:
 47:     CLOSE
```

```
/* :MERGE-ACCIDENT-FILES */

 48:        FILE (ACC1),
 49:        FILE (ACC2),
 50:        FILE (TEMPD);
 51:     PUT FILE (SYSPRINT) SKIP(2) EDIT ('DETAIL FILE MERGED') (A);
 52:     OPEN
 53:        FILE (TEMPD) TITLE ('TEMPFILE'),
 54:        FILE (ACC2) OUTPUT TITLE ('ACIDENT');
 55:     ON ENDFILE (TEMPD) GOTO CLOSED2;

 56: LOOPD3:
 57:     READ FILE (TEMPD) INTO (D2);
 58:     WRITE FILE (ACC2) FROM (D2) KEYFROM (KD2);
 59:     GOTO LOOPD3;

 60: CLOSED2:
 61:     CLOSE
 62:        FILE (ACC2),
 63:        FILE (TEMPD);
 64:     PUT FILE (SYSPRINT) SKIP EDIT ('DETAIL FILE COPIED') (A);


 65: /*** VEHICLE FILE ***/

 66:     OPEN
 67:        FILE (ACC1) TITLE ('ACCVEHM'),
 68:        FILE (ACC2) TITLE ('ACCVEH'),
 69:        FILE (TEMPV) OUTPUT TITLE ('TEMPFILE');
 70:     ON ENDFILE (ACC1) GOTO EOFV1;
 71:     ON ENDFILE (ACC2) GOTO EOFV2;
 72:     READ FILE (ACC2) INTO (V2);

 73: LOOPV:
 74:     READ FILE (ACC1) INTO (V1);
 75:     DO WHILE (KV2<KV1);
 76:        WRITE FILE (TEMPV) FROM (V2);
 77:        READ FILE (ACC2) INTO (V2);
 78:        END;
 79:     DO WHILE (KV1=KV2);
 80:        READ FILE (ACC2) INTO (V2);
 81:        END;
 82:     WRITE FILE (TEMPV) FROM (V1);
 83:     GOTO LOOPV;

 84: EOFV1:
 85:     ON ENDFILE (ACC2) GOTO CLOSEV;

 86: LOOPV1:
 87:     WRITE FILE (TEMPV) FROM (V2);
 88:     READ FILE (ACC2) INTO (V2);
 89:     GOTO LOOPV1;

 90: EOFV2:
 91:     ON ENDFILE (ACC1) GOTO CLOSEV;

 92: LOOPV2:
 93:     WRITE FILE (TEMPV) FROM (V1);
 94:     READ FILE (ACC1) INTO (V1);
 95:     GOTO LOOPV2;
```

```
/* :MERGE-ACCIDENT-FILES */


 96: CLOSEV:
 97:    CLOSE
 98:       FILE (ACC1),
 99:       FILE (ACC2),
100:       FILE (TEMPV);
101:    PUT FILE (SYSPRINT) SKIP(2) EDIT ('VEHICLE FILE MERGED') (A);
102:    OPEN
103:       FILE (TEMPV) TITLE ('TEMPFILE'),
104:       FILE (ACC2) OUTPUT TITLE ('ACCVEH');
105:    ON ENDFILE (TEMPV) GOTO CLOSEV2;

106: LOOPV3:
107:    READ FILE (TEMPV) INTO (V2);
108:    WRITE FILE (ACC2) FROM (V2) KEYFROM (KV2A);
109:    GOTO LOOPV3;

110: CLOSEV2:
111:    CLOSE
112:       FILE (ACC2),
113:       FILE (TEMPV);
114:    PUT FILE (SYSPRINT) SKIP EDIT ('VEHICLE FILE COPIED') (A);

115: END MERGE;
```

PRINT-MEMOS --

```
Member Name . . . . . PUA
Language . . . . . PL/I
Subroutines . . . . . PRINTX1
Files . . . . . SYSPRINT  -- IBM messages
                PRNTMEMO  -- PUA output
                MEMOS     -- Memos file
Instruction . . . . . 1 - 3  "PUA"
```

PRINT-MEMOS prints the accident memos. The data for these memos is stripped from the accident cards and written into a memos file at load time by LOAD-ACCIDENT-DATA.

The PUA program listing follows:

```
/* :PRINT-MEMOS */

 1: /* :PRINT-MEMOS */

 2: PMEMO:  PROCEDURE (PARM) OPTIONS (MAIN);

 3: /* INSTRUCTION AND PRINT ROUTINE */
 4: DECLARE
 5:    PARM_CHAR(100),
 6:    INSTR CHAR(80) EXT,
 7:    PRINTER CHAR(132) EXT,
 8:    PRINTX ENTRY (PIC'Z'),
 9:   (INIT,EXIT) ENTRY,
10:  PRNT FILE INT RECORD OUTPUT ENV (F(1330,133) CTLASA);

11: /* MEMO FILE */
12: DECLARE
13:    1  M BASED (PTR_M),
14:       2  ACC_# CHAR(12),
15:       2 (MO,DAY,YR) CHAR(2),
16:       2  CNTY PIC'ZZ',
17:       2  LOCN CHAR(12),
18:       2  #_VEH PIC'ZZ',
19:       2  SEV PIC'Z',
20:       2  NAME CHAR(20),
21:       2  INIT(2) CHAR(1),
22:       2  LICENSE CHAR(17),
23:       2  STATE CHAR(2),
24:       2  CHARGE CHAR(6),
25:    MEMOS FILE INT RECORD;

26: /* OTHER VARIABLES */
27: DECLARE
28:    SEV(3) CHAR(15) STATIC INIT
29:       ('FATAL','INJURY','PROPERTY DAMAGE');


30: /***** INITIALIZATION *****/

31:    OPEN FILE (PRNT) TITLE ('PRNTMEMO');
32:    CALL INIT (PARM);
33:    OPEN FILE (MEMOS);
34:    ON ENDFILE (MEMOS) GOTO DONE;


35: /***** MAIN LOOP *****/

36: LOOP:
37:    READ FILE (MEMOS) SET (PTR_M);
38:    PRINTER = (10)' ' ||
39:       'MONTANA HIGHWAY PATROL - ACCIDENT MEMO';
40:    CALL PRINTX (9);
41:    PRINTER = (5)' ' ||
42:       'NAME:  ' || M.INIT(1) || '. ' || M.INIT(2) || '. ' || M.NAME
43:    CALL PRINTX (3);
44:    PRINTER = (5)' ' ||
45:       'DL #:  ' || M.LICENSE;
46:    CALL PRINTX (1);
47:    PRINTER = (5)' ' ||
48:       'STATE:  ' || M.STATE;
```

```
/* :PRINT-MEMOS */

    CALL PRINTX (1);
    PRINTER = (5)' ' ||
       'CHARGE:  ' || M.CHARGE;
    CALL PRINTX (1);
    PRINTER = (5)' ' ||
       'ACCIDENT NUMBER:  ' || M.ACC_#;
    CALL PRINTX (3);
    PRINTER = (5)' ' ||
       'LOCATION:  ' || M.LOCN;
    CALL PRINTX (1);
    PRINTER = (5)' ' ||
       'COUNTY:  ' || M.CNTY;
    CALL PRINTX (1);
    PRINTER = (5)' ' ||
       'DATE:  ' || M.MO || '/' || M.DAY || '/' || M.YR;
    CALL PRINTX (1);
    PRINTER = (5)' ' ||
       'NUMBER OF VEHICLES:  ' || M.#_VEH;
    CALL PRINTX (1);
    PRINTER = (5)' ' ||
       'SEVERITY:  ' || SEV(M.SEV);
    CALL PRINTX (1);
    GOTO LOOP;


/***** TERMINATION *****/

DONE:
    CLOSE FILE (MEMOS);
    CALL EXIT (PARM);

END PMEMO;
```

FORM-16 --

Member Name	NS	
Language	PL/I	
Subroutines	PRINTX1	
	FORM16A	
	FORM16B	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- FORM16 output
	ACIDENT	-- Detail file
	ACCVEH	-- Vehicle file
	CITYTBL	-- Table of city names
Instruction	1 - 2	"NS"
	24 - 31	Starting date (mm/dd/yy)
	32 - 39	Ending date (mm/dd/yy)
	40 - 57	Location

FORM-16 produces the National Safety Council Form 16 report for statewide accidents. Only reportable accidents (those with $250 or more damage to the property of one person, or involving injuries or fatalities) are included when LOCATION=ALL. All accidents, regardless of extensiveness of damage, are included in municipal runs. Form 16 consists of 21 separate tables (two of which are not produced by FORM-16 due to lack of data). Because of the large number of tables, FORM-16 is quite a large program, and has been designed as a planned overlay structure. The root program, FORM16, sets up an area of core for storing numbers, and initializes the area to zeroes. It loads phase FORM16A into core to calculate the tables values. It then loads phase FORM16B into the area occupied formerly by FORM16A for the purpose of printing the report. To conserve storage, FORM16A does not calculate totals; these are calculated as the values are printed. There are 936 array elements (each declared decimal fixed (5,0)) in form 16. Twice this amount is set up, in order that a printout of both rural and urban accidents may be obtained with only one pass through the file. After printing both the rural and urban summaries, FORM16B adds the two together, and prints a summary including both. STORAGE, an array with 1872 decimal fixed (5,0) elements, is set up by the root program and initialized to zero. The array is passed to each of the other two phases when they are invoked. This phase divides the array in half, using elements 1-936 for rural accidents, and 937-1872 for urban accidents.

A structure of arrays, ARRAY, contains all of the arrays for the tables, and is based on pointer PTR_STOR. This pointer is set to the address of STORAGE(1) when processing rural accidents, and to the address of STORAGE(937) when processing urban accidents. Within the structure ARRAY are the arrays used for the tables. These arrays are named A_n, where n is the number of the table on the form. For example, A_11 is the array for table 11, and A_1B is the array for Table 1B. FORM16A is the computational phase. It reads the accident detail file, searching for accidents within the time range specified on the command. Each time one is found, the vehicle records are retrieved from the accident vehicle file, and the appropriate values added into the arrays. FORM16B is the print-out phase. It prints the tables, using the values computed by FORM16A, and totals calculated within FORM16B. FORM16B has a large number of variables declared in order to print the summaries. An effort has been made to be consistent in the naming of these variables. Variables A_n are the arrays set up by FORM16A. The numbers for each line of output are placed first into a computational structure, allowing the numbers in the structures to be manipulated without undue data conversions; these computational structures are named C_n. After a line is set up in a computational structure, the values are converted to character format, and a description added, in an output structure O_n. Totals are kept in totals structures, T_n. HDG_n contain headings for table n.

TABLE 1-A:

Table 1-A (see Figure 2-IV-6) is a breakdown of the number of accidents by first harmful event, injury severity, and roadway-related location. A 12x3x2 array is utilized in calculating the values. Each accident is shown in exactly one element of the array. The first harmful event provides the first subscript. This item is coded as a number between 1 and 11; any other value is included under 12 (unknown). The second subscript indicates the injury severity of the accident: "1" for fatal, "2" for non-fatal injury, and "3" for property damage accidents. The final subscript indicates the roadway-related location: "1" for on roadway, and "2" for off

1A. TYPE OF MOTOR-VEHICLE ACCIDENT		Number of Accidents											
		Total				On Roadway				Off Roadway			
		Total	Fatal	Nonfatal Injury	Property Damage	Total	Fatal	Nonfatal Injury	Property Damage	Total	Fatal	Nonfatal Injury	Property Damage
Non-collision	1. Overturning												
	2. Other noncollision												
Collision involving:	3. Pedestrian												
	4. MV in transport												
	5. MV on other roadway												
	6. Parked MV												
	7. Railway train												
	8. Pedalcyclist												
	9. Animal												
	10. Fixed object												
	11. Other object												
	12.												
	Totals												

1B. TYPE OF MOTOR-VEHICLE ACCIDENT		Number of Persons					
		Total Killed	Total Injured	Incapacitating Injury	Non-Incapac. Evident Injury	Possible Injury	No Injury
Non-collision	1. Overturning						
	2. Other noncollision						
Collision involving:	3. Pedestrian						
	4. MV in transport						
	5. MV on other roadway						
	6. Parked MV						
	7. Railway train						
	8. Pedalcyclist						
	9. Animal						
	10. Fixed object						
	11. Other object						
	12.						
	Totals						

Figure 2-IV-6. Form 16--Tables 1-A and 1-B.

roadway. If a value other than 1 or 2 appears in the roadway-related location field, the accident is shown as on roadway.

TABLE 1-B:

Table 1-B (see Figure 2-IV-6) is a breakdown of the number of persons involved in accidents by first harmful event and by severity. A 12x5 array is used for calculations. Each person involved in an accident (as pedestrian, driver, or passenger) is included in exactly one element of the array. The first subscript is the first harmful event, as is calculated in Table 1-A. The second subscript indicates the person's injury severity: "1" for fatal, "2" for incapacitating injury, "3" for non-incapacitating injury, "4" for possible injury, and "5" for no injury.

TABLE 2-A:

Table 2-A (see Figure 2-IV-7) shows the number of accidents, fatalities, and injuries by first harmful event and roadway-related location. A_2A, a 12x3x2 array, is used in calculating the values. The first subscript is the first harmful event, and is calculated as in Table 1-A. The last subscript is the roadway-related location, also calculated as in Table 1-A. The second subscript is used for differentiating accidents, fatalities, and injuries. For example, an accident with first harmful event 4 (motor vehicle in transport) occurring on roadway is shown in element (4,1,1). The number of fatalities in this accident is shown in element (4,2,1), while the number of injuries is shown in element (4,3,1).

TABLE 2-B:

Table 2-B (see Figure 2-IV-7) requires data pertaining to mileage rates which is not available in the databank. Hence, this table is not produced.

2A. TYPE OF MOTOR-VEHICLE ACCIDENT		Total						On Roadway					
		This Year To Date			Same Period Last Year			This Year To Date			Same Period Last Year		
		All Accidents	Persons Killed	Persons Injured	All Accidents	Persons Killed	Persons Injured	All Accidents	Persons Killed	Persons Injured	All Accidents	Persons Killed	Persons Injured
Non-collision	1. Overturning												
	2. Other noncollision												
Collision involving:	3. Pedestrian												
	4. MV in transport												
	5. MV on other roadway												
	6. Parked MV												
	7. Railway train												
	8. Pedalcyclist												
	9. Animal												
	10. Fixed object												
	11. Other object												
	12.												
	Totals												

2B. MILEAGE RATES	This Year To Date	Last Year Same Period	Percent Change
1. Motor vehicle traffic deaths			%
2. Estimated motor vehicle mileage traveled (millions)			%
3. Death rate per 100,000,000 vehicle-miles			%
4. Fatal accident rate per 100,000,000 vehicle-miles			%
5.			%
6.			%

Figure 2-IV-7. Form 16--Tables 2-A and 2-B.

TABLE 3-A:

Table 3-A (see Figure 2-IV-8) gives a breakdown of accidents occurring in cities of population 2500 and greater, by city size, injury severity, and roadway-related location. Array A_3A1 (5x3x2) is used for calculating the values for the table. In addition, the number of fatalities and injuries are shown by city size. A_3A2 (5x2) is used for these calculations. To obtain the city populations, a table of city names and populations stored in library HIS.TABLES (member name CITYTBL) is used. During program initialization, this table is read, and the population codes (a number from 1 to 7 in column 55 of the city table records) stored in an array. Codes 1 and 2 indicate cities of less than 2500; hence, only cities with codes 3-7 are included. The codes are reduced by 2 (giving a range from 1 to 5), and used as the first subscript in both arrays. The code thus indicate:

1 2500-5000
2 5000-10,000
3 10,000-25,000
4 25,000-50,000
5 50,000-100,000

The second subscript of A_3A1 is the injury severity; the third is the roadway-related location. These are calculated as in Table 1-A. The second subscript of A-3A2 differentiates between fatalities and injuries. Hence, element (1,1) gives the number of fatalities in cities of size 2500-5000, and element (1,2) gives the corresponding number of injuries.

TABLES 3-B and 3-C:

Tables 3-B and 3-C (see Figure 2-IV-8) give breakdowns of municipal and rural accidents, respectively, by class of trafficway, injury severity, and roadway-related location. A_3BC1

3. LOCATION	Number of Accidents												Number of Persons	
	Total				On Roadway				Off Roadway				Total	
	Total	Fatal	Nonfatal Injury	Property Damage	Total	Fatal	Nonfatal Injury	Property Damage	Total	Fatal	Nonfatal Injury	Property Damage	Killed	Injured
3A. Municipalities and Incorporated Townships														
1 2,500 to 5,000														
2 5,000 to 10,000														
3. 10,000 to 25,000														
4. 25,000 to 50,000														
5. 50,000 to 100,000														
6. 100,000 to 250,000														
7. 250,000 or more														
8.														
9.														
Total														
3B. URBAN: See ANS D16.1—1970, Manual on Classification of Motor Vehicle Traffic Accidents, for definition.														
1. Interstate system														
2. Other full control access														
3. Other U.S. route numbered														
4. Other state numbered														
5. Other major arterial														
6 County roads														
7. Local streets														
8. Other trafficways														
9. Not stated														
Total Urban														
3C. RURAL: All locations other than those above.														
1. Interstate system														
2. Other full control access														
3. Other U.S. route numbered														
4 Other state numbered														
5. Other major arterial														
6. County roads														
7. Local streets														
8. Other trafficways														
9. Not stated														
Total Rural														

Figure 2-IV-8. Form 16--Tables 3-A, 3-B, and 3-C.

(2x9x3x2) is used for the computations. In addition, A_3BC2 (2x9x2) is used for computing the number of fatalities and injuries by municipal/rural location and class of trafficway. The first element of each array is the municipal/rural location. All elements having a 1 in this position are destined for Table 3-B (municipal accidents); those having a 2 are for Table 3-C. The second subscripts indicate the class of trafficway. This is coded in the files as a number from 1-8; any other code is shown in category 9 (not stated). The third and fourth subscripts of A_3BC1 correspond to the second and third subscripts of A_3A1. The final subscript of A_3BC2 corresponds to the final subscript of A_3A2.

TABLE 4:

This table (see Figure 2-IV-9) shows the number of fatalities and injuries by age, sex, and whether pedestrain, pedalcyclist, or other. Array A_4 (12x2x2x3) is used for the computations. Each person injured or killed is shown exactly once in the table. The first subscript is the age:

1	0-4	7	35-44
2	5-9	8	45-54
3	10-14	9	55-64
4	15-19	10	65-74
5	20-24	11	75 & Older
6	25-34	12	Not Stated

If the age is coded as 0, the person is shown in category 12. The second subscript differentiates the fatalities and injuries. Hence, element (4,1,1,2) gives the number of pedestrians, male, from 15 to 19, that were killed. Element (4,2,1,2) gives the corresponding number of injuries. The third subscript is the sex: "1" for male, and "2" for female. If the sex is not given, the person is shown as female. The fourth subscript is the type:

1 All but pedestrians and pedalcyclists
2 Pedestrians
3 Pedalcyclists

4. AGE OF CASUALTY	Number of Persons Killed									Number of Persons Injured								
	Total Killed			Pedestrians			Pedalcyclist			Total Injured			Pedestrians			Pedalcyclist		
	Total	Male	Female	Total	Male	Female	Total	Male	Female	Total	Male	Female	Total	Male	Female	Total	Male	Female
1. 0 to 4																		
2. 5 to 9																		
3. 10 to 14																		
4. 15 to 19																		
5. 20 to 24																		
6. 25 to 34																		
7. 35 to 44																		
8. 45 to 54																		
9. 55 to 64																		
10. 65 to 74																		
11. 75 & older																		
12. Not stated																		
Totals																		

6. PEDESTRIAN ACTIONS BY AGE	Pedestrians Killed	Ages of Pedestrians Killed and Injured									
		Total	0 to 4	5 to 9	10 to 14	15 to 19	20 to 24	25 to 44	45 to 64	65 & Older	Not Stated
1a. Crossing at intersection or in crosswalk											
1b. Crossing not at intersection or in crosswalk											
2a. Walking in roadway—with traffic											
2b. Same—against traffic											
3. Standing in roadway											
4. Pushing or working on vehicle in roadway											
5. Other working in roadway											
6. Playing in roadway											
7. Other in roadway											
8. Not in roadway											
9. Not stated											
Totals											

Figure 2-IV-9. Form 16--Tables 4 and 6.

TABLES 5-A, 5-B, 5-C, and 5-D:

These tables (see Figure 2-IV-10) provide a directional analysis of the accidents. Each accident appear in exactly one of these four tables. Parts 5-A and 5-B show two-vehicle accidents (5-A showing those at intersections, 5-B those not at intersections). Part 5-C shows pedestrian accidents. Part 5-D shows all other accidents.

TABLE 5-A:

An accident is shown in Table 5-A if all of the following are true:

Number of Vehicles is 2,
Number of Pedestrians is 0, and
Junction-Related Location is 1 or 2.

A_5A (9x3) is used for calculating the values for Table 5-A. The second subscript gives the injury severity (1 for fatal, 2 for non-fatal injury, and 3 for property damage) of the accident. The first subscript is the directional analysis, depending upon the collision type (TYPE) and the vehicle intents (INT1 and INT2), and has the values:

1 TYPE=3 and
INT1 and INT2 any value.

2 TYPE=2 or 5,
INT1=1, 2, or 6, and
INT2=1, 2, or 6.

3 TYPE=2 or 5,
INT1=1, 2, or 6, and
INT2=3, 4, or 5.

4 TYPE=2 or 5, and
INT1=7-11 or INT2=7-11.

5 TYPE=2 or 5--all others.

5A. TWO MOTOR VEHICLE ACC.		Total	Fatal Accidents	Injury Accidents	Property Damage Acc.
At Intersection	1. Entering at angle				
	2a. From same direction—both going straight				
	2b. Same—one turn, one straight				
	2c. Same—one stopped				
	2d. Same—all others				
	3a. From opposite direction—both going straight				
	3b. Same—one left turn, one straight				
	3c. Same—all others				
	4. Not stated				
	Totals				

5B. TWO MOTOR VEHICLE ACC.		Total	Fatal Accidents	Injury Accidents	Property Damage Acc.
Not at Intersection	1. Going opposite direction—both moving				
	2. Going same direction—both moving				
	3a. One car parked				
	3b. One car stopped in traffic				
	4a. One car entering parked position				
	4b. One car leaving parked position				
	5a. One car entering driveway access				
	5b. One car leaving driveway access				
	6. All others				
	7. Not stated				
	Totals				

5C. PEDESTRIAN ACCIDENTS	All Pedestrian Accidents	Fatal Accidents			Non-Fatal Injury Accidents		
		Total	At intersection, Intersec.—related	Driveway access, Nonjunction	Total	At intersection, Intersec.—related	Driveway access Nonjunction
1. Car going straight							
2. Car turning right							
3. Car turning left							
4. Car backing							
5. All others							
Totals							

5D. ALL OTHER ACCIDENTS			Total	Fatal Accidents	Injury Accidents	Property Damage Acc.
At Intersection	Collision with:	1. Other road vehicle, or railway train				
		2. Fixed object				
		3. Other object or animal				
	4. Overturning					
	5. Other noncollision					
Not At Intersection	Collision with:	6. Other road vehicle, or railway train				
		7. Fixed object				
		8. Other object or animal				
	9. Overturning					
	10. Other noncollision					
11. Not stated						
Totals						

Figure 2-IV-10. Form 16--Tables 5-A through 5-D.

6 TYPE=1 or 4,
INT1=1, and
INT2=1.

7 TYPE=1 or 4,
INT1=1, and
INT2=4.

8 TYPE=1 or 4--all others.

9 All others.

The subscript in A_5A corresponds to the line numbers in Table 5-A as follows:

1	1	6	3a
2	2a	7	3b
3	2b	8	3c
4	2c	9	4
5	2d		

TABLE 5-B:

An accident is included in Table 5-B if all of the following are true:

Number of Vehicles is 2,
Number of Pedestrians is 0, and
Junction-Related Location is 0 or 3.

A_5B (9x3) is used for calculating Table 5-B. The second subscript is the injury severity (see part 5-A). The first subscript is the directional analysis, depending upon the collision type (TYPE), the vehicle intents (INT1 and INT2), and the junction-related location (JCT). This subscript has the values:

1 JCT=0,
TYPE=1 or 4,
INT1=1, 3, 4, or 5, and
INT2=1, 3, 4, or 5.

2 JCT=0,
TYPE=2 or 5,
INT1=1-6, and
INT2=1-6.

3 JCT=0,
TYPE=any value, and
Either INT1=11 or INT2=11.

4 JCT=0,
TYPE=any value, and
Either INT1=10 or INT2=10.

5 Not presently used.

6 JCT=0,
TYPE=any value, and
Either INT1=8 or INT2=8.

7 JCT=3, and
TYPE, INT1, INT2=any values.

8 All others--TYPE not zero.

9 All others--TYPE=0.

The first subscript in A_5B corresponds to the line numbers in Table 5-B as follows:

1	1	6	4b
2	2	7	5a and 5b
3	3a	8	6
4	3b	9	7
5	4a		

Item 5 (line 4a) cannot be calculated, as no code is available for entering parked position. Lines 5a and 5b are lumped into one category, as it is not possible to distinguish, for driveway access accidents, whether a vehicle was entering or leaving the driveway access. The categories have been combined into a "driveway access" category, rather than showing the accidents under category 8 (all others).

TABLE 5-C:

An accident is included in Table 5-C if:

Number of Vehicles is 1, and
Number of Pedestrians is 1.

Array A-5C (5x3x2) is used to compute Table 5-C. The second subscript is the injury severity, as in Table 5-A. The third subscript indicates the junction-related location (1 for intersection or intersection-related, 2 for driveway access or non-junction). The first subscript is the directional analysis, dependent only upon the driver's intent (INT1), and has the values:

1 INT1=1
2 INT1=3
3 INT1=4
4 INT1=9
5 All others

The subscript values correspond directly to the line numbers of the Table.

TABLE 5-D:

An accident is included in Table 5-D if:

Number of Vehicles + Number of Pedestrians is not 2 (all accidents not shown in Tables 5-A through 5-C).

Array 5-D (11x3) is used for the computations. The second subscript is the injury severity, as in part 5-A. The first subscript is the directional analysis, dependent upon the first harmful event (FHE) and junction-related location (JCT). This subscript may take on the values:

1 JCT=1 or 2, and
FHE not 1, 2, 9, 10, or 11.

2 JCT=1 or 2, and
FHE=10.

3 JCT=1 or 2, and
FHE=9 or 11.

4 JCT=1 or 2, and
FHE=1.

5 JCT=1 or 2, and
FHE=2.

6-10 Same as 1-5, except JCT not 1 or 2.

11 FHE=0.

These values correspond directly to the line number of the table.

TABLE 6:

Table 6 (see Figure 2-IV-9) shows the number of pedestrians killed by action, and the number of pedestrians killed or injured by age and action. One-dimensional array A_6_1 (11) contains the number of pedestrians killed. Array A_6_2 (11x9) contains the number of pedestrians killed or injured, by action and age. The first subscript gives the action. This is the pedestrian intent (if coded as 0 or larger than 10, the pedestrian is included in class 11). The second subscript of A_6_2 is the age:

1 0-4
2 5-9
3 10-14
4 15-19
5 20-24
6 25-44
7 45-64
8 65 & Older
9 Not Stated

If the age is not coded, the pedestrian is included in class 9 rather than class 1 as zero.

TABLE 7:

Table 7 (see Figure 2-IV-11) presents a summary of drivers of vehicles (other than properly parked vehicles) involved in

7. AGE OF DRIVER	All Accidents	Fatal Accidents	Injury Accidents
1. 15 & younger			
2. 16			
3. 17			
4. 18 to 19			
5. 20 to 24			
6. 25 to 34			
7. 35 to 44			
8. 45 to 54			
9. 55 to 64			
10. 65 to 74			
11. 75 & older			
12. Not stated			
Totals			

8. SEX OF DRIVER	All Accidents	Fatal Accidents	Injury Accidents
1. Male			
2. Female			
3. Not stated			
Totals			

9. RESIDENCE OF DRIVER	All Accidents	Fatal Accidents	Injury Accidents
1. Local resident			
2. Residing elsewhere in state			
3. Non-resident of state			
4. Not stated			
Totals			

10. CONTRIBUTING CIRCUMSTANCES	All Accidents	Fatal Accidents	Injury Accidents
1. Speed too fast			
2. Failed to yield right of way			
3. Passed stop sign			
4. Disregarded traffic signal			
5. Drove left of center			
6. Improper overtaking			
7. Followed too closely			
8. Made improper turn			
9. Had been drinking			
10. Other improper driving			
11. Mechanical defect			
12. Other			
Totals			

12. ROAD SURFACE CONDITION	All Accidents	Fatal Accidents	Injury Accidents
1. Dry			
2. Wet			
3. Snowy or Icy			
4. Other			
5. Not stated			
Totals			

13. LIGHT CONDITION	All Accidents	Fatal Accidents	Injury Accidents
1. Daylight			
2. Dawn or dusk			
3. Darkness			
4. Not stated			
Totals			

11. TYPE OF VEHICLE	All Accidents	Fatal Accidents	Injury Accidents
1. Passenger car			
2. Passenger car and trailer			
3. Truck or truck tractor			
4. Truck tractor and semi-trailer			
5. Other truck combination			
6. Farm tractor and/or farm equip.			
7. Taxicab			
8. Bus			
9. School bus			
10. Motorcycle			
11. Motor scooter or motor bicycle			
12. Other			
13. Not stated			
Totals			
Special vehicles included above			
14. Emergency (including privately owned)			
15. Military vehicles			
16. Other publicly owned vehicles			

14. MANNER OF TWO MOTOR VEHICLE COLLISION	All Accidents	Fatal Accidents	Injury Accidents
1. Head on			
2. Rear end			
3. Angle			
4. Sideswipe—Meeting			
5. Sideswipe—Passing			
6. Backed into			
7. Not stated			
Totals			

Figure 2-IV-11. Form 16--Tables 7 through 14.

accidents, by age and injury severity of the accident. Array A_7 (12x3) is used to contain the values calculated. The first subscript is the driver's age:

1 15 & Younger
2 16
3 17
4 18-19
5 20-24
6 25-34
7 35-44
8 45-54
9 55-64
10 65-74
11 75 & Older
12 Not Stated

The second subscript is the injury severity: 1 for all accidents, 2 for fatal accidents, and 3 for injury accidents. Hence, property damage accidents appear only in category 1, fatal accidents appear in both 1 and 2, and injury accidents appear in both 1 and 3.

TABLE 8:

Table 8 (see Figure 2-IV-11) shows the same drivers in Table 7, broken down by sex rather than age. Array A_8 (3x3) is used for computations. The first subscript is the sex:

1 Male
2 Female
3 Not Stated

The second subscript is used as in Table 7.

TABLE 9:

Table 9 (see Figure 2-IV-11) requires information on driver's residence which is not coded on Montana accident reports. Hence, this table is not produced.

TABLE 10:

Table 10 (see Figure 2-IV-11) shows accidents by contributing circumstances. The contributing circumstances may be obtained from the possible violations coded on the vehicle records, and from the actual charges. A_10 is 12x3, the second subscript being identical to that in Table 7. The first subscript is the contributing circumstances:

1 PV = 3 or CHARGE = 5130H through 5190H

2 PV = 4 or CHARGE = 5350H, 5380H, 5400H through 5451H, 5453H, 5460H

3 PV = 5 or CHARGE = 5580H, 5610H

4 CHARGE = 5050H through 5064H, 5080H, 5570H, 5620H through 5640H

5 CHARGE = 5200H or 5210H

6 CHARGE = 5220H through 5260H

7 CHARGE = 5290H

8 CHARGE = 5330H through 5340H, 5360H through 5362H, 5380H through 5391H

9 PV = 1 or CHARGE = 5110H

10 Other PV not zero

11 Mech. Defect non-zero or CHARGE = 7000H through 7010H, 7020, 7261H

12 Any other contributing circumstance non-zero

TABLE 11:

Table 11 (see Figure 2-IV-11) shows all vehicles by body and trailer style. Its array, A_11, is dimensioned 16x3. The second subscript is used as in part 7. The first subscript depends on the body style (BODY) and trailer style (TRLR):

1	BODY=1 and TRLR=0
2	BODY=1 and TRLR not 0
3	BODY=5,6 and TRLR=0
4	BODY=5,6 and TRLR=5
5	BODY=5,6 all other
6	BODY=10
7	not used
8	BODY=3
9	BODY=4
10	BODY=8
11	not used
12	all not shown in 1-11 or 13-16, BODY=9
13	BODY=0
14	BODY=9
15	not used
16	not used

NOTE: Ambulance (BODY=9) shown under both 12 and 14.

TABLE 12:

Table 12 (see Figure 2-IV-11) lists accidents by road surface condition. The array, A_12, is 5x3, the second subscript used as in part 7. The first subscript is the road condition, and is related to the road condition coded in the file by:

1	1
2	2
3	3 or 4
4	5
5	0 or larger than 5

TABLE 13:

This table (see Figure 2-IV-11) presents a breakdown by light condition of all accidents. The array, A_13, is 4x3, the second subscript used as in part 7. The first subscript gives the light condition, and is related to the light condition coded in the file by:

1	1
2	2
3	3 or 4
4	0 or larger than 4

TABLE 14:

Table 14 (see Figure 2-IV-11) presents two-vehicle accidents by collision type. The array A_14 is 7x3, the second subscript used as in part 7. The first subscript is the collision type, and has the same value as that coded in the file (if coded as zero, or larger than 7, the accident is shown in class 7).

The NS, FORM16A and FORM16B program listing follow:

```
OFORM16:  PROCEDURE (PARM) OPTIONS (MAIN);

   1: FORM16:  PROCEDURE (PARM) OPTIONS (MAIN);

   2: DECLARE
   3:    PARM CHAR(100),
   4:    INSTR CHAR(80) EXT,
   5:    STORAGE(1872) DEC FIXED (5,0) STATIC;

   6:    STORAGE = 0;
   7:    INSTR = PARM;
   8:    CALL FORM16A (STORAGE,ERROR_RETURN);
   9:    CALL INIT (PARM);
  10:    CALL FORM16B (STORAGE);

  11: ERROR_RETURN:
  12:    CALL EXIT (PARM);

  13: END FORM16;
```

```
)FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);


  1: FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);

  2: /* INSTRUCTION */
  3: DECLARE
  4:    INSTR CHAR(80) EXT,
  5:    START_DATE CHAR(8) DEF INSTR POS(24),
  6:    START_MONTH PIC'ZZ' DEF INSTR POS(24),
  7:    START_DAY PIC'ZZ' DEF INSTR POS(27),
  8:    START_YEAR PIC'ZZ' DEF INSTR POS(30),
  9:    END_DATE CHAR(8) DEF INSTR POS(32),
 10:    END_MONTH PIC'ZZ' DEF INSTR POS(32),
 11:    END_DAY PIC'ZZ' DEF INSTR POS(35),
 12:    END_YEAR PIC'ZZ' DEF INSTR POS(38),
 13:    LOCATION CHAR(18) DEF INSTR POS(40);

 14: /* DETAIL RECORD */
 15: DECLARE
 16:    1  DET BASED (PTRA),
 17:       2  DUM1 CHAR(1),
 18:       2  KEY CHAR(12),
 19:       2  OCCURRED,
 20:          3 (MONTH,DAY,YEAR,HOUR,MIN) DEC FIXED (3,0),
 21:       2  DUM2 CHAR(20),
 22:       2 (CITY_#,CNTY_#) DEC FIXED (3,0),
 23:       2  MILEPOST CHAR(12),
 24:       2  BLOCK_A,
 25:          3 (FIRST_EVENT,FIRST_OBJ) DEC FIXED (3,0),
 26:          3 (INJ_SEV,DAM_SEV,TRAFFICWAY,RDY_REL,JCT_REL)
 27:             DEC FIXED (1,0),
 28:       2  BLOCK_B,
 29:          3 (#_VEH,#_PED,#_FAT,#_INJ) DEC FIXED (3,0),
 30:          3 (WEATHER,ROAD,LIGHT) DEC FIXED (1,0),
 31:       2  BLOCK_C,
 32:          3 (CONTROLS,OTH_DAM_TYPE) DEC FIXED (3,0),
 33:          3 (OTH_DAM_SEV,OTHER_DAM_OWNER) DEC FIXED (1,0),
 34:          3  SPEED DEC FIXED (3,0),
 35:          3  ENG_STUDY CHAR(1),
 36:          3  ANAL(2) DEC FIXED (3,0),
 37:          3  COLL_TYPE DEC FIXED (1,0),
 38:       2 (REPORTABLE,INVESTIGATED) CHAR(1),
 39:    ACIDENT FILE INT RECORD KEYED ENV (INDEXED GENKEY);

 40: /* VEHICLE RECORD */
 41: DECLARE
 42:    1  VEH BASED (PTRV),
 43:       2  DUM1 CHAR(1),
 44:       2  KEY CHAR(12),
 45:       2  VEH_PED CHAR(1),
 46:       2  VEH_# PIC'99',
 47:       2  LAST_NAME CHAR(22),
 48:       2  DRIV_LICENSE CHAR(17),
 49:       2  STATE CHAR(2),
 50:       2  BIRTHDAY CHAR(6),
 51:       2  RE_EXAM CHAR(1),
 52:       2  DUM2 CHAR(1),
 53:       2  CHARGE CHAR(5),
 54:       2  SUMMONS CHAR(6),
```

```
)FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);

  55:        2  CONTR_CIRC(5) DEC FIXED (1,0),
  56:        2  PASS(6),
  57:           3  ALCOHOL DEC FIXED (1,0),
  58:           3  SEX CHAR(1),
  59:           3  INJ DEC FIXED (1,0),
  60:           3  AGE DEC FIXED (3,0),
  61:        2 (VEH_YEAR,INTENT,BODY) DEC FIXED (3,0),
  62:        2  TRAILER DEC FIXED (1,0),
  63:        2  INTERSTATE_TRAF CHAR(1),
  64:        2  VEH_ID CHAR(15),
  65:        2  REPORTABLE CHAR(1),
  66:        2  VEH_DAM DEC FIXED (1,0),
  67:     ACCVEH FILE INT RECORD KEYED ENV (INDEXED GENKEY);

  68: /* STORAGE OF VALUES */
  69: DECLARE
  70:     STORAGE(1872) DEC FIXED (5,0),
  71:     1  ARRAY BASED (PTR_STOR),
  72:        2 (A_1A(12,3,2),
  73:           A_1B(12,5),
  74:           A_2A(12,3,2),
  75:           A_3A1(5,3,2),
  76:           A_3A2(5,2),
  77:           A_3BC1(2,9,3,2),
  78:           A_3BC2(2,9,2),
  79:           A_4(12,2,2,3),
  80:           A_5A(9,3),
  81:           A_5B(9,3),
  82:           A_5C(5,3,2),
  83:           A_5D(11,3),
  84:           A_6_1(11),
  85:           A_6_2(11,9),
  86:           A_7(12,3),
  87:           A_8(3,3),
  88:           A_10(12,3),
  89:           A_11(16,3),
  90:           A_12(5,3),
  91:           A_13(4,3),
  92:           A_14(7,3))
  93:           DEC FIXED (5,0);

  94: /* OTHER VARIABLES */
  95: DECLARE
  96:    (DTE,SDTE,EDTE) DEC FIXED (7,0),
  97:    (EVNT,INJ1,INJ2,INJ7,ON_OFF,INT(2),AGE,I,J,K,L) DEC FIXED (3,0),
  98:     C1 CHAR(1),
  99:     C5 CHAR(5),
 100:    (PTR_RURAL,PTR_URBAN) PTR,
 101:     ERROR_RETURN LABEL,
 102:     TABLE FILE INT RECORD,
 103:     1  CITY_RECORD BASED (PTR),
 104:        2  DUM1 CHAR(36),
 105:        2  NAME CHAR(18),
 106:        2  POP PIC'Z',
 107:     CITY DEC FIXED (3,0),
 108:     POP(0:126) DEC FIXED (1,0) STATIC,
 109:     FLAG CHAR(1);
```

```
DFORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);


110: /***** INITIALIZATION *****/
111:    ON ERROR GOTO ERROR_RETURN;
112:    PTR_RURAL = ADDR(STORAGE(1));
113:    PTR_URBAN = ADDR(STORAGE(937));
114:    IF LOCATION=' ' THEN LOCATION = 'ALL';

115:    /* READ CITY TABLE */
116:    OPEN FILE (TABLE) INPUT RECORD TITLE ('CITYTBL');
117:    CITY = 0;
118:    DO I=1 TO 126;
119:       READ FILE (TABLE) SET (PTR);
120:       POP(I) = CITY_RECORD.POP;
121:       IF LOCATION=CITY_RECORD.NAME THEN CITY = I;
122:       END;
123:    CLOSE FILE (TABLE);
124:    IF LOCATION¬='ALL' & CITY=0 THEN DO;
125:       PUT FILE (SYSPRINT) SKIP EDIT
126:         ('CITY SPECIFIED IS NOT KNOWN') (A);
127:       GOTO ERROR_RETURN;
128:       END;
129:    POP(0) = 0;
130:    OPEN
131:       FILE (ACIDENT),
132:       FILE (ACCVEH);
133:    ON KEY (ACIDENT);
134:    READ FILE (ACIDENT) SET (PTRA) KEY (START_YEAR);

135:    /* CALCULATE STARTING AND ENDING DATES */
136:    SDTE = 10000*START_YEAR + 100*START_MONTH + START_DAY;
137:    EDTE = 10000*END_YEAR + 100*END_MONTH + END_DAY;
138:    ON ENDFILE (ACIDENT) GOTO DONE;
139:    ON ENDFILE (ACCVEH) VEH.KEY = 'XX';


140: /***** EXECUTION LOOP *****/

141: LOOP:
142:    DTE = 10000*DET.YEAR + 100*DET.MONTH + DET.DAY;
143:    IF DTE>EDTE | DET.YEAR<START_YEAR THEN GOTO READ_DATA;
144:    IF DET.REPORTABLE¬='X' & CITY=0 THEN GOTO READ_DATA;
145:    IF CITY¬=0 & DET.CITY_#¬=CITY THEN GOTO READ_DATA;

146:    /* SET ARRAY INDICATORS */
147:    IF CITY=0 THEN
148:       IF DET.CITY_#=0
149:          THEN PTR_STOR = PTR_RURAL;
150:          ELSE PTR_STOR = PTR_URBAN;
151:       ELSE IF DET.REPORTABLE='X'
152:          THEN PTR_STOR = PTR_RURAL;
153:          ELSE PTR_STOR = PTR_URBAN;
154:    EVNT = DET.FIRST_EVENT;
155:    IF EVNT=0 | EVNT>11 THEN EVNT = 11;
156:    IF DET.#_FAT¬=0
157:       THEN DO;
158:          INJ1 = 1;
159:          INJ7 = 2;
```

```
)FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);

160:          END;
161:       ELSE IF DET.#_INJ¬=0
162:          THEN DO;
163:             INJ1 = 2;
164:             INJ7 = 3;
165:             END;
166:          ELSE DO;
167:             INJ1 = 3;
168:             INJ7 = 1;
169:             END;
170:    ON_OFF = DET.RDY_REL;
171:    IF ON_OFF=0 | ON_OFF>2 THEN ON_OFF = 1;

172:    /* ONLY 2-A SHOWS "THIS YEAR TO DATE" */
173:    IF DTE<SDTE THEN GOTO READ_DATA;

174:    /* PART 1-A */
175:    A_1A(EVNT,INJ1,ON_OFF) = A_1A(EVNT,INJ1,ON_OFF) + 1;

176:    /* PART 2-A */
177:    A_2A(EVNT,1,ON_OFF) = A_2A(EVNT,1,ON_OFF) + 1;
178:    A_2A(EVNT,2,ON_OFF) = A_2A(EVNT,2,ON_OFF) + DET.#_FAT;
179:    A_2A(EVNT,3,ON_OFF) = A_2A(EVNT,3,ON_OFF) + DET.#_INJ;

180:    /* PART 3-A */
181:    IF POP(DET.CITY_#)>=3 THEN DO;
182:       I = POP(DET.CITY_#) - 2;
183:       A_3A1(I,INJ1,ON_OFF) = A_3A1(I,INJ1,ON_OFF) + 1;
184:       A_3A2(I,1) = A_3A2(I,1) + DET.#_INJ;
185:       A_3A2(I,2) = A_3A2(I,2) + DET.#_FAT;
186:       END;

187:    /* PARTS 3-B AND 3-C */
188:    IF DET.CITY_#¬=0
189:       THEN I = 1;
190:       ELSE I = 2;
191:    J = DET.TRAFFICWAY;
192:    IF J=0 THEN J = 9;
193:    A_3BC1(I,J,INJ1,ON_OFF) = A_3BC1(I,J,INJ1,ON_OFF) + 1;
194:    A_3BC2(I,J,1) = A_3BC2(I,J,1) + DET.#_INJ;
195:    A_3BC2(I,J,2) = A_3BC2(I,J,2) + DET.#_FAT;

196:    /* GET VEHICLE RECORDS */
197:    READ FILE (ACCVEH) SET (PTRV) KEY (DET.KEY);
198:    DO L=1 TO 100 WHILE (DET.KEY=VEH.KEY);

199:       /* PARTS 1-B AND 4 */
200:       DO J=1 TO 6;
201:          C1 = VEH.PASS(J).SEX;
202:          INJ2 = VEH.PASS(J).INJ;
203:          AGE = VEH.PASS(J).AGE;
204:          IF C1='M' | C1='F' | AGE¬=0 | INJ2¬=0 THEN DO;
205:             IF INJ2=0 | INJ2>5 THEN INJ2 = 5;
206:             A_1B(EVNT,INJ2) = A_1B(EVNT,INJ2) + 1;
207:             IF INJ2<5 THEN DO;
208:                IF AGE=0 THEN AGE = 12;
209:                   ELSE IF AGE>=75 THEN AGE = 11;
210:                   ELSE IF AGE>=25 THEN AGE = (AGE+5)/10 + 3;
```

```
)FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);

211:                    ELSE AGE = AGE/5 + 1;
212:                IF INJ2¬=1 THEN INJ2 = 2;
213:                IF C1='M'
214:                   THEN I = 1;
215:                   ELSE I = 2;
216:                IF VEH.VEH_PED='B' THEN K = 2;
217:                   ELSE IF VEH.BODY=13 THEN K = 3;
218:                   ELSE K = 1;
219:                A_4(AGE,INJ2,I,K) = A_4(AGE,INJ2,I,K) + 1;
220:                END;
221:             END;
222:          END;

223:       /* SAVE INTENTS FOR PART 5 IF NECESSARY */
224:       IF DET.#_VEH+DET.#_PED=2 & VEH.VEH_PED¬='C'
225:          THEN INT(L) = VEH.INTENT;

226:       /* PART 6 -- PEDESTRIANS */
227:       IF VEH.VEH_PED='B' & VEH.PASS(1).INJ¬=0 THEN DO;
228:          I = VEH.INTENT;
229:          IF I=0 | I>11 THEN I = 11;
230:          AGE = VEH.PASS(1).AGE;
231:          IF AGE=0 THEN AGE = 9;
232:             ELSE IF AGE>=65 THEN AGE = 8;
233:             ELSE IF AGE>=45 THEN AGE = 7;
234:             ELSE IF AGE>=25 THEN AGE = 6;
235:             ELSE AGE = AGE/5 + 1;
236:          IF VEH.PASS(1).INJ=1 THEN A_6_1(I) = A_6_1(I) + 1;
237:          A_6_2(I,AGE) = A_6_2(I,AGE) + 1;
238:          END;

239:       /* PARTS 7, 8, 10, AND 11 -- VEHICLES */
240:       IF VEH.VEH_PED='A' THEN DO;

241:          /* 7 AND 8 EXCLUDE PROPERLY PARKED VEHICLES */
242:          IF VEH.INTENT¬=8 & VEH.INTENT¬=11 THEN DO;
243:             AGE = VEH.PASS(1).AGE;
244:             IF AGE=0 THEN AGE = 12;
245:                ELSE IF AGE>=75 THEN AGE = 11;
246:                ELSE IF AGE>=25 THEN AGE = (AGE-5)/10 + 4;
247:                ELSE IF AGE>=20 THEN AGE = 5;
248:                ELSE IF AGE>=18 THEN AGE = 4;
249:                ELSE IF AGE=17 THEN AGE = 3;
250:                ELSE IF AGE=16 THEN AGE = 2;
251:                ELSE AGE = 1;
252:             A_7(AGE,1) = A_7(AGE,1) + 1;
253:             IF INJ7¬=1 THEN A_7(AGE,INJ7) = A_7(AGE,INJ7) + 1;
254:             IF VEH.PASS(1).SEX='M' THEN I = 1;
255:                ELSE IF VEH.PASS(1).SEX='F' THEN I = 2;
256:                ELSE I = 3;
257:             A_8(I,1) = A_8(I,1) + 1;
258:             IF INJ7¬=1 THEN A_8(I,INJ7) = A_8(I,INJ7) + 1;
259:             END;
260:          K = VEH.CONTR_CIRC(5);
261:          C5 = VEH.CHARGE;
262:          IF K=3 THEN I = 1;
263:             ELSE IF K=4 THEN I = 2;
264:             ELSE IF K=5 THEN I = 3;
```

```
)FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);

265:              ELSE IF K=1 THEN I = 9;
266:              ELSE IF K¬=0 THEN I = 10;
267:              ELSE I = 0;
268:           IF K=0 THEN DO;
269:              IF C5>='5130H' & C5<='5190H' THEN I = 1;
270:              IF C5='5350H' | C5='5380H' | C5>='5400H' & C5<='5460H' &
271:                 C5¬='5452H' THEN I = 2;
272:              IF C5='5580H' | C5='5610H' THEN I = 3;
273:              IF C5>='5050H' & C5<='5064H' | C5='5080H' | C5='5570H' |
274:                 C5>='5620H' & C5<='5640H' THEN I = 4;
275:              IF C5='5200H' | C5='5210H' THEN I = 5;
276:              IF C5>='5220H' & C5<='5260H' THEN I = 6;
277:              IF C5='5290H' THEN I = 7;
278:              IF C5>='5330H' & C5<='5391H' & C5¬='5350H' &
279:                 C5¬='5363H' THEN I = 8;
280:              IF C5='5110H' THEN I = 9;
281:              END;
282:           IF I¬=0 THEN DO;
283:              A_10(I,1) = A_10(I,1) + 1;
284:              IF INJ7¬=1 THEN A_10(I,INJ7) = A_10(I,INJ7) + 1;
285:              END;
286:           IF VEH.CONTR_CIRC(4)¬=0 | C5>='7000H' & C5<='7010H' |
287:              C5='07020' | C5='7261H' THEN DO;
288:              A_10(11,1) = A_10(11,1) + 1;
289:              IF INJ7¬=1 THEN A_10(11,INJ7) = A_10(11,INJ7) + 1;
290:              END;
291:           IF VEH.CONTR_CIRC(1)¬=0 | VEH.CONTR_CIRC(2)¬=0 |
292:              VEH.CONTR_CIRC(3)¬=0 THEN DO;
293:              A_10(12,1) = A_10(12,1) + 1;
294:              IF INJ7¬=1 THEN A_10(12,INJ7) = A_10(12,INJ7) + 1;
295:              END;
296:           I = VEH.BODY;
297:           J = VEH.TRAILER;
298:           K = 12;
299:           IF I=1
300:              THEN IF J=0
301:                 THEN K = 1;
302:                 ELSE K = 2;
303:           IF I=5 | I=6
304:              THEN IF J=0 THEN K = 3;
305:              ELSE IF J=5 THEN K = 4;
306:              ELSE K = 5;
307:           IF I=10 THEN K = 6;
308:           IF I=3 THEN K = 8;
309:           IF I=4 THEN K = 9;
310:           IF I=8 THEN K = 10;
311:           IF I=0 THEN K = 13;
312:           A_11(K,1) = A_11(K,1) + 1;
313:           IF INJ7¬=1 THEN A_11(K,INJ7) = A_11(K,INJ7) + 1;
314:           IF I=9 THEN DO;
315:              A_11(14,1) = A_11(14,1) + 1;
316:              IF INJ7¬=1 THEN A_11(14,INJ7) = A_11(14,INJ7) + 1;
317:              END;
318:           END;
319:        READ FILE (ACCVEH) SET (PTRV);
320:        END;

321:     /* PART 5 */
```

```
)FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);

322:     IF DET.#_VEH=2 & DET.#_PED=0
323:        THEN IF DET.JCT_REL=1 | DET.JCT_REL=2
324:           THEN DO;
325:              IF INT(1)>INT(2) THEN DO; /* FORCE INT(1)<=INT(2) */
326:                 J = INT(1);
327:                 INT(1) = INT(2);
328:                 INT(2) = J;
329:                 END;
330:              I = DET.COLL_TYPE;
331:              IF I=3 THEN J = 1;
332:                 ELSE IF I=2 | I=5 THEN DO;
333:                    IF (INT(1)<=2 | INT(1)=6) & (INT(2)<=2 | INT(2)=6)
334:                       THEN J = 2;
335:                    ELSE IF INT(1)<=2 & INT(2)>=3 & INT(2)<=5 |
336:                       INT(1)>=3 & INT(1)<=5 & INT(2)=6
337:                       THEN J = 3;
338:                    ELSE IF INT(1)>=7 | INT(2)>=7 THEN J = 4;
339:                    ELSE J = 5;
340:                    END;
341:                 ELSE IF I=1 | I=4 THEN DO;
342:                    IF INT(1)=1 & INT(2)=1 THEN J = 6;
343:                    ELSE IF INT(1)=1 & INT(2)=4 THEN J = 7;
344:                    ELSE J = 8;
345:                    END;
346:                 ELSE J = 9;
347:              A_5A(J,INJ1) = A_5A(J,INJ1) + 1;
348:              END;
349:           ELSE DO;
350:              IF INT(1)>INT(2) THEN DO; /* FORCE INT(1)<=INT(2) */
351:                 J = INT(1);
352:                 INT(1) = INT(2);
353:                 INT(2) = J;
354:                 END;
355:              I = DET.COLL_TYPE;
356:              IF DET.JCT_REL=3 THEN J = 7;
357:              ELSE IF (I=1|I=4) & INT(1)<=5 & INT(1)¬=2 & INT(2)<=5 &
358:                 INT(2)¬=2 THEN J = 1;
359:              ELSE IF (I=2 | I=5) & INT(1)<=6 & INT(2)<=6 THEN J = 2;
360:              ELSE IF INT(1)=11 | INT(2)=11 THEN J = 3;
361:              ELSE IF INT(1)=10 | INT(2)=10 THEN J = 4;
362:              ELSE IF (INT(1)=8 | INT(2)=8) & DET.JCT_REL=0 THEN J = 6;
363:              ELSE IF I¬=0 THEN J = 8;
364:              ELSE J = 9;
365:              A_5B(J,INJ1) = A_5B(J,INJ1) + 1;
366:              END;
367:        ELSE IF DET.#_VEH=1 & DET.#_PED=1 THEN DO;
368:           IF INT(1)=1 THEN J = 1;
369:              ELSE IF INT(1)=3 THEN J = 2;
370:              ELSE IF INT(1)=4 THEN J = 3;
371:              ELSE IF INT(1)=9 THEN J = 4;
372:              ELSE J = 5;
373:           IF DET.JCT_REL=1 | DET.JCT_REL=2
374:              THEN K = 1;
375:              ELSE K = 2;
376:          A_5C(J,INJ1,K) = A_5C(J,INJ1,K) + 1;
377:           END;
378:        ELSE DO;
379:           IF EVNT=1 THEN J = 4;
```

```
)FORM16A:  PROCEDURE (STORAGE,ERROR_RETURN);

380:              ELSE IF EVNT=2 THEN J = 5;
381:              ELSE IF EVNT=10 THEN J = 2;
382:              ELSE IF EVNT>=9 THEN J = 3;
383:              ELSE J = 1;
384:           IF DET.JCT_REL=0 | DET.JCT_REL=3 THEN J = J + 5;
385:           IF EVNT=0 THEN J = 11;
386:           A_5D(J,INJ1) = A_5D(J,INJ1) + 1;
387:           END;

388:     /* PART 12 */
389:     I = DET.ROAD;
390:     IF I=4 THEN I = 3;
391:     IF I=5 THEN I = 4;
392:     IF I=0 | I>5 THEN I = 5;
393:     A_12(I,1) = A_12(I,1) + 1;
394:     IF INJ7¬=1 THEN A_12(I,INJ7) = A_12(I,INJ7) + 1;

395:     /* PART 13 */
396:     I = DET.LIGHT;
397:     IF I=4 THEN I = 3;
398:     IF I=0 | I>4 THEN I = 4;
399:     A_13(I,1) = A_13(I,1) + 1;
400:     IF INJ7¬=1 THEN A_13(I,INJ7) = A_13(I,INJ7) + 1;

401:     /* PART 14 */
402:     IF DET.#_VEH=2 THEN DO;
403:     I = DET.COLL_TYPE;
404:     IF I=0 | I>7 THEN I = 7;
405:     A_14(I,1) = A_14(I,1) + 1;
406:     IF INJ7¬=1 THEN A_14(I,INJ7) = A_14(I,INJ7) + 1;
407:     END;

408: READ_DATA:
409:     READ FILE (ACIDENT) SET (PTRA);
410:     GOTO LOOP;

411: DONE:
412:     CLOSE FILE (ACIDENT);
413:     CLOSE FILE (ACCVEH);

414: END FORM16A;
```

```
FORM16B:  PROCEDURE (STORAGE);

 1: FORM16B:  PROCEDURE (STORAGE);

 2: /* PRINT ROUTINE */
 3: DECLARE
 4:   (PRINTER,HEADING(9)) CHAR(132) EXT,
 5:    PRINTX ENTRY (PIC'Z'),
 6:    PRINTXA ENTRY (PIC'Z',PIC'ZZ'),
 7:    INSTR CHAR(80) EXT,
 8:    LOCATION CHAR(18) DEF INSTR POS(40),
 9:    #_HDGS PIC'Z' DEF INSTR POS(72);

10: /* ARRAYS */

11: DECLARE
12:    STORAGE(1872) DEC FIXED (5,0),
13:    1  ARRAY BASED (PTR_STOR),
14:       2 (A_1A(12,3,2),
15:          A_1B(12,5),
16:          A_2A(12,3,2),
17:          A_3A1(5,3,2),
18:          A_3A2(5,2),
19:          A_3BC1(2,9,3,2),
20:          A_3BC2(2,9,2),
21:          A_4(12,2,2,3),
22:          A_5A(9,3),
23:          A_5B(9,3),
24:          A_5C(5,3,2),
25:          A_5D(11,3),
26:          A_6_1(11),
27:          A_6_2(11,9),
28:          A_7(12,3),
29:          A_8(3,3),
30:          A_10(12,3),
31:          A_11(16,3),
32:          A_12(5,3),
33:          A_13(4,3),
34:          A_14(7,3))
35:          DEC FIXED (5,0);

36: /* OUTPUT STRUCTURES */
37: DECLARE
38:    OUT CHAR(136) STATIC,
39:    1  O_1A DEF OUT POS(4),
40:       2  DESCR CHAR(25),
41:       2  VAL,
42:          3  R1,
43:             4 (T,N(3)) PIC'(8)Z',
44:          3  R2,
45:             4  T PIC'(12)Z',
46:             4  N(3) PIC'(8)Z',
47:          3  R3 LIKE O_1A.R2,
48:    1  O_1B DEF OUT POS(12),
49:       2  DESCR CHAR(25),
50:       2  N(6) PIC'(14)Z',
51:    1  O_2A DEF OUT POS(4),
52:       2  DESCR CHAR(25),
53:       2  R1,
54:          3  N(3) PIC'(8)Z',
```

```
FORM16B:  PROCEDURE (STORAGE);

 55:         2  DUMMY CHAR(29),
 56:         2  R2 LIKE O_2A.R1,
 57:     1   O_3 DEF OUT,
 58:         2  DESCR CHAR(27),
 59:         2  VAL,
 60:            3  R1,
 61:               4 (T,N(3)) PIC'(8)Z',
 62:            3 (R2,R3) LIKE O_3.R1,
 63:            3  #_FAT PIC'(4)Z',
 64:            3  #_INJ PIC'(5)Z',
 65:     1   O_4 DEF OUT,
 66:         2  DESCR CHAR(15),
 67:         2  VAL,
 68:            3  R1,
 69:               4 (T,N(2)) PIC'(6)Z',
 70:            3  R2,
 71:               4  T PIC'(7)Z',
 72:               4  N(2) PIC'(6)Z',
 73:            3  R3 LIKE O_4.R2,
 74:            3  R4,
 75:               4  T PIC'(11)Z',
 76:               4  N(2) PIC'(6)Z',
 77:            3 (R5,R6) LIKE O_4.R2,
 78:     1   O_5A DEF OUT POS(27),
 79:         2  DESCR CHAR(39),
 80:         2  VAL,
 81:            3 (T,N(3)) PIC'(10)Z',
 82:     1   O_5C DEF OUT POS(14),
 83:         2  DESCR CHAR(25),
 84:         2  VAL,
 85:            3  T PIC'(8)Z(5)B',
 86:            3  R1,
 87:               4 (T,N(2)) PIC'(10)Z',
 88:            3  R2,
 89:               4  T PIC'(15)Z',
 90:               4  N(2) PIC'(10)Z',
 91:     1   O_6 DEF OUT,
 92:         2  DESCR CHAR(36),
 93:         2  VAL,
 94:            3  #_FAT PIC'ZZZZZ',
 95:            3  TOT PIC'(10)Z',
 96:            3  N(9) PIC'(9)Z',
 97:     1   O_7 DEF OUT POS(33),
 98:         2  DESCR CHAR(45),
 99:         2  N(3) PIC'(10)Z',
100:     Z PIC'Z';

101: /* CALCULATION STRUCTURES */
102: DECLARE
103:     1   C_1A STATIC,
104:         2  R1,
105:            3 (T,N(3)) DEC FIXED (7,0),
106:         2 (R2,R3) LIKE C_1A.R1,
107:     1   C_1B STATIC,
108:         2  N(6) DEC FIXED (7,0),
109:     1   C_2A STATIC,
110:         2  R1,
111:            3  N(3) DEC FIXED (7,0),
```

```
FORM16B:  PROCEDURE (STORAGE);

112:        2  R2 LIKE C_2A.R1,
113:     1  C_3 STATIC,
114:        2  R1,
115:           3 (T,N(3)) DEC FIXED (7,0),
116:        2 (R2,R3) LIKE C_3.R1,
117:        2 (#_FAT,#_INJ) DEC FIXED (5,0),
118:     1  C_4 STATIC,
119:        2  R1,
120:           3 (T,N(2)) DEC FIXED (7,0),
121:        2 (R2,R3,R4,R5,R6) LIKE C_4.R1,
122:     1  C_5A STATIC,
123:        2 (T,N(3)) DEC FIXED (7,3),
124:     1  C_5C STATIC,
125:        2  T DEC FIXED (7,0),
126:        2  R1,
127:           3 (T,N(2)) DEC FIXED (7,0),
128:        2  R2 LIKE C_5C.R1,
129:     1  C_6 STATIC,
130:        2 (#_FAT,TOT,N(9)) DEC FIXED (5,0),
131:     1  C_7 STATIC,
132:        2  N(3) DEC FIXED (7,0);

133: /* ARRAYS FOR TOTALS */
134: DECLARE
135:   (T_1A(3,2),
136:    T_1B(5),
137:    T_2A(3,2),
138:    T_3A1(3,2),
139:    T_3A2(2),
140:    T_4(2,2,3),
141:    T_5A(3),
142:    T_5C(3,2),
143:    T_6_1,
144:    T_6_2(9),
145:    T_7(3)) DEC FIXED (7,0) STATIC;

146: /* DESCRIPTION ARRAYS */
147: DECLARE
148:    EVENT(13) CHAR(25) STATIC INIT (
149:       ' 1.  OVERTURNING',
150:       ' 2.  OTHER NONCOLLISION',
151:       ' 3.  PEDESTRIAN',
152:       ' 4.  MV IN TRANSPORT',
153:       ' 5.  MV ON OTHER ROADWAY',
154:       ' 6.  PARKED MV',
155:       ' 7.  RAILWAY TRAIN',
156:       ' 8.  PEDALCYCLIST',
157:       ' 9.  ANIMAL',
158:       '10.  FIXED OBJECT',
159:       '11.  OTHER OBJECT',
160:       '12.  UNKNOWN',
161:       '       TOTALS'),
162:    ROAD(6) CHAR(16) STATIC INIT (
163:       '1.  DRY',
164:       '2.  WET',
165:       '3.  SNOWY OR ICY',
166:       '4.  OTHER',
167:       '5.  NOT STATED',
```

```
FORM168:  PROCEDURE (STORAGE);

168:        '    TOTALS'),
169:     LIGHT(6) CHAR(16) STATIC INIT (
170:        '1.  DAYLIGHT',
171:        '2.  DAWN OR DUSK',
172:        '3.  DARKNESS',
173:        '4.  NOT STATED',
174:        '    TOTALS',' '),
175:     TYPE(8) CHAR(21) STATIC INIT (
176:        '1.  HEAD ON',
177:        '2.  REAR END',
178:        '3.  ANGLE',
179:        '4.  SIDESWIPE-MEETING',
180:        '5.  SIDESWIPE-PASSING',
181:        '6.  BACKED INTO',
182:        '7.  NOT STATED',
183:        '    TOTALS'),
184:     TRAF(10) CHAR(27) STATIC INIT (
185:        '1.  INTERSTATE SYSTEM',
186:        '2.  OTHER CONTROL ACCESS',
187:        '3.  OTHER US ROUTE NUMBERED',
188:        '4.  OTHER STATE NUMBERED',
189:        '5.  OTHER MAJOR ARTERIAL',
190:        '6.  COUNTY ROADS',
191:        '7.  LOCAL STREETS',
192:        '8.  OTHER TRAFFICWAYS',
193:        '9.  NOT STATED',
194:        '      TOTAL URBAN'),
195:     RUR_URB(2) CHAR(10) STATIC INIT (
196:        '3B.  URBAN',
197:        '3C.  RURAL'),
198:     POP(8) CHAR(22) STATIC INIT (
199:        '1.    2,500 TO   5,000',
200:        '2.    5,000 TO  10,000',
201:        '3.   10,000 TO  25,000',
202:        '4.   25,000 TO  50,000',
203:        '5.   50,000 TO 100,000',
204:        '6.  100,000 TO 250,000',
205:        '7.  250,000 AND OVER',
206:        '      TOTAL'),
207:     AGE(13) CHAR(15) STATIC INIT (
208:        ' 1.  0 TO 4',
209:        ' 2.  5 TO 9',
210:        ' 3.  10 TO 14',
211:        ' 4.  15 TO 19',
212:        ' 5.  20 TO 24',
213:        ' 6.  25 TO 34',
214:        ' 7.  35 TO 44',
215:        ' 8.  45 TO 54',
216:        ' 9.  55 TO 64',
217:        '10.  65 TO 74',
218:        '11.  75 & OLDER',
219:        '12.  NOT STATED',
220:        '     TOTALS'),
221:     DIREC_5A(10) CHAR(38) STATIC INIT (
222:        '1.   ENTERING AT ANGLE',
223:        '2A.  SAME DIRECTION--BOTH STRAIGHT',
224:        '2B.  SAME--ONE TURNING, ONE STRAIGHT',
225:        '2C.  SAME--ONE STOPPED',
```

```
FORM16B:  PROCEDURE (STORAGE);

226:        '2D.  SAME--ALL OTHERS',
227:        '3A.  OPPOSITE DIRECTION--BOTH STRAIGHT',
228:        '3B.  SAME--ONE LEFT TURN,ONE STRAIGHT',
229:        '3C.  SAME--ALL OTHERS',
230:        '4.   NOT STATED',
231:        '     TOTALS'),
232:     DIREC_5B(10) CHAR(38) STATIC INIT (
233:        '1.   OPPOSITE DIRECTION--BOTH MOVING',
234:        '2.   SAME DIRECTION--BOTH MOVING',
235:        '3A.  ONE CAR PARKED',
236:        '3B.  ONE CAR STOPPED IN TRAFFIC',
237:        '4A.  ONE CAR ENTERING PARKED POSITION',
238:        '4B.  ONE CAR LEAVING PARKED POSITION',
239:        '5.   DRIVEWAY ACCESS',
240:        '6.   ALL OTHERS',
241:        '7.   NOT STATED',
242:        '     TOTALS'),
243:     DIREC_5C(6) CHAR(22) STATIC INIT (
244:        '1.  CAR GOING STRAIGHT',
245:        '2.  CAR TURNING RIGHT',
246:        '3.  CAR TURNING LEFT',
247:        '4.  CAR BACKING',
248:        '5.  ALL OTHERS',
249:        '     TOTALS'),
250:     DIREC_5D(12) CHAR(29) STATIC INIT (
251:        ' 1.  OTHER ROAD VEHICLE/TRAIN',
252:        ' 2.  FIXED OBJECT',
253:        ' 3.  OTHER OBJECT OR ANIMAL',
254:        ' 4.  OVERTURNING',
255:        ' 5.  OTHER NONCOLLISION',
256:        ' 6.  OTHER ROAD VEHICLE/TRAIN',
257:        ' 7.  FIXED OBJECT',
258:        ' 8.  OTHER OBJECT OR ANIMAL',
259:        ' 9.  OVERTURNING',
260:        '10.  OTHER NONCOLLISION',
261:        '11.  NOT STATED',
262:        '     TOTALS'),
263:     PED_ACT(12) CHAR(36) STATIC INIT (
264:        '1A.  AT INTERSECTN OR IN CROSSWALK',
265:        '1B.  NOT AT INTERSECTN OR CROSSWALK',
266:        '2A.  WALKING IN RDWAY--WITH TRAFFIC',
267:        '2B.  SAME--AGAINST TRAFFIC',
268:        '3.   STANDING IN ROADWAY',
269:        '4.   PUSHING/WORKING ON VEH IN RDWAY',
270:        '5.   OTHER WORKING IN ROADWAY',
271:        '6.   PLAYING IN ROADWAY',
272:        '7.   OTHER IN ROADWAY',
273:        '8.   NOT IN ROADWAY',
274:        '9.   NOT STATED',
275:        '     TOTALS'),
276:     DRIV_AGE(13) CHAR(17) STATIC INIT (
277:        ' 1.  15 & YOUNGER',
278:        ' 2.  16',
279:        ' 3.  17',
280:        ' 4.  18 TO 19',
281:        ' 5.  20 TO 24',
282:        ' 6.  25 TO 34',
283:        ' 7.  35 TO 44',
```

```
FORM16B:  PROCEDURE (STORAGE);

284:        ' 8.  45 TO 54',
285:        ' 9.  55 TO 64',
286:        '10.  65 TO 74',
287:        '11.  75 & OLDER',
288:        '12.  NOT STATED',
289:        '     TOTALS'),
290:     SEX(4) CHAR(14) STATIC INIT (
291:        '1.  MALE',
292:        '2.  FEMALE',
293:        '3.  NOT STATED',
294:        '    TOTALS'),
295:     CIRC(13) CHAR(33) STATIC INIT (
296:        ' 1.  SPEED TOO FAST',
297:        ' 2.  FAILED TO YIELD RIGHT OF WAY',
298:        ' 3.  PASSED STOP SIGN',
299:        ' 4.  DISREGARDED TRAFFIC SIGNAL',
300:        ' 5.  DROVE LEFT OF CENTER',
301:        ' 6.  IMPROPER OVERTAKING',
302:        ' 7.  FOLLOWED TOO CLOSELY',
303:        ' 8.  MADE IMPROPER TURN',
304:        ' 9.  HAD BEEN DRINKING',
305:        '10.  OTHER IMPROPER DRIVING',
306:        '11.  MECHANICAL DEFECT',
307:        '12.  OTHER',
308:        '     TOTALS'),
309:     VEH_TYPE(17) CHAR(33) STATIC INIT (
310:        ' 1.  PASSENGER CAR',
311:        ' 2.  PASSENGER CAR & TRAILER',
312:        ' 3.  TRUCK OR TRUCK TRACTOR',
313:        ' 4.  TRUCK TRACTOR & SEMI-TRAILER',
314:        ' 5.  OTHER TRUCK COMBINATION',
315:        ' 6.  FARM TRACTOR AND/OR EQUIP',
316:        ' 7.  TAXICAB',
317:        ' 8.  BUS',
318:        ' 9.  SCHOOL BUS',
319:        '10.  MOTORCYCLE',
320:        '11.  MOTOR SCOOTER/MOTOR BIKE',
321:        '12.  OTHER',
322:        '13.  NOT STATED',
323:        '     TOTALS',
324:        '14.  EMERGENCY (INCL. PRIVATE)',
325:        '15.  MILITARY VEHICLES',
326:        '16.  OTHER PUBLICLY OWNED VEHS');

327: /* SUMMARY HEADINGS */
328: DECLARE

329:     HDG_1A(4) CHAR(132) STATIC INIT (
330:        '1A.  TYPE OF                                  ********************************
331: ******* NUMBER OF ACCIDENTS ***************************************
332:        '     MOTOR-VEHICLE                            ************ TOTAL ***********
333:    ********* ON ROADWAY *********                      ******** OFF ROADWAY **********'
334:        '     ACCIDENT.
335:                   NONFATAL  PROP                                  NONFATAL  PROP
336:        '                                                           NONFATAL  PROP',
337:     TOTAL    FATAL   INJURY  DAMAGE                  TOTAL   FATAL   INJURY  DAMAGE
                                                          TOTAL   FATAL   INJURY  DAMAGE'

338:     HDG_1B(4) CHAR(110) STATIC INIT (
```

```
FORM16B:  PROCEDURE (STORAGE);

339:       '1B.  TYPE OF                                  ************************** N
340: UMBER OF PERSONS **********************',
341:       '     MOTOR-VEHICLE
342:          NON-INCAPAC.',
343:       '     ACCIDENT.                          TOTAL        TOTAL    INCAPAC
344: ITATING    EVIDENT       POSSIBLE          NO',
345:       '                                      KILLED       INJURED       INJ
346: URY          INJURY        INJURY         INJURY'),

347:    HDG_2A(4) CHAR(132) STATIC INIT (
348:       '2A.  TYPE OF                        ********************* TOTAL *******
349: ***************  ****************** ON ROADWAY ********************',
350:       '     MOTOR-VEHICLE                 THIS TIME PERIOD        SAME PER
351: IOD LAST YEAR       THIS TIME PERIOD          SAME PERIOD LAST YEAR  ',
352:       '     ACCIDENT.                  ALL    PERSONS PERSONS    ALL ~
353: PERSONS PERSONS     ALL    PERSONS PERSONS    ALL    PERSONS PERSONS',
354:       '                                ACCIDENTS  KILLED INJURED ACCIDENTS
355:  KILLED INJURED  ACCIDENTS  KILLED INJURED ACCIDENTS  KILLED INJURED'),

356:    HDG_3(4) CHAR(132) STATIC INIT (
357:       '3A.  MUNICIPALITIES AND      ********************************** NUM
358: BER OF ACCIDENTS ****************************************  NUMBER ',
359:       '     INCORPORATED             *********** TOTAL ************  ***
360: ****** ON ROADWAY **********  ********* OFF ROADWAY *******    OF    ',
361:       '     TOWNSHIPS                                    NONFATAL  PROP
362:          NONFATAL  PROP                          NONFATAL  PROP  PERSONS',
363:       '                                TOTAL   FATAL  INJURY  DAMAGE   TO
364: TAL   FATAL  INJURY  DAMAGE    TOTAL   FATAL  INJURY  DAMAGE FAT INJ'),

365:    HDG_4(3) CHAR(132) STATIC INIT (
366:       '4.  AGE OF      *************** NUMBER OF PERSONS KILLED *******
367: ********     ************** NUMBER OF PERSONS INJURED **************',
368:       '    CASUALTY.     TOTAL KILLED        PEDESTRIANS       PEDALCYC
369: LIST          TOTAL INJURED       PEDESTRIANS       PEDALCYCLIST',
370:       '                TOTAL MALE FEMALE  TOTAL MALE FEMALE  TOTAL MALE
371:  FEMALE      TOTAL MALE FEMALE  TOTAL MALE FEMALE  TOTAL MALE FEMALE'),

372:    HDG_5A(3) CHAR(80) STATIC INIT (
373:       '5A.  TWO MOTOR-VEHICLE ACCIDENTS.
374:        PROPERTY',
375:       '                                                    FATAL    IN
376: JURY    DAMAGE',
377:       '     AT INTERSECTION.                        TOTAL  ACCIDENTS ACC
378: IDENTS ACCIDENTS'),

379:    HDG_5C(3) CHAR(105) STATIC INIT (
380:       '5C.  PEDESTRIAN              ALL          ***** FATAL ACCIDENTS
381:  *****        NON-FATAL INJURY ACCIDENTS ',
382:       '     ACCIDENTS.          PEDESTRIAN                          DRI
383: VEWAY,                        DRIVEWAY,',
384:       '                          ACCIDENTS        TOTAL  INTERSECTN  NO
385: NJCT          TOTAL  INTERSECTN  NONJCT  '),
386:    HDG_6(3) CHAR(132) STATIC INIT (
387:       '6.  PEDESTRIAN ACTIONS BY AGE.                  ******************
388: ***** AGES OF PEDESTRIANS KILLED AND INJURED **********************',
389:       '
390:                                                  65 &     NOT',
391:       '                                  KILLED   TOTAL      0-4
```

```
FORM16B:  PROCEDURE (STORAGE);

392:   5-9    10-14    15-19    20-24    25-44    45-64    OLDER   STATED')


393: /***** INITIALIZATION *****/

394:    HEADING = ' ';
395:    #_HDGS = 4;
396:    DO I=1 TO 18;
397:       IF SUBSTR(LOCATION,I,1)='-' THEN SUBSTR(LOCATION,I,1) = ' ';
398:       END;
399:    SUBSTR(HEADING(1),50,34) = 'SUMMARY OF MOTOR VEHICLE ACCIDENTS';
400:    IF LOCATION='ALL'
401:       THEN SUBSTR(HEADING(3),60,15) = 'RURAL ACCIDENTS';
402:       ELSE SUBSTR(HEADING(3),54,28) = 'LEGALLY REPORTABLE ACCIDENTS';
403:    PTR_STOR = ADDR(STORAGE(1));
404:    M1 = 1;
405:    M2 = 1;


406: /***** MAIN EXECUTION LOOP *****/

407: LOOP:
408:    IF LOCATION='ALL'
409:       THEN PRINTER = 'STATEWIDE ACCIDENTS';
410:       ELSE PRINTER = 'CITY OF ' || LOCATION;
411:    CALL PRINTX (9);
412:    PRINTER = 'REPORTING PERIOD FROM ' || SUBSTR(INSTR,24,8) ||
413:       ' TO ' || SUBSTR(INSTR,32,8);
414:    CALL PRINTX (2);
415:    PRINTER = 'LEGALLY REPORTABLE ACCIDENTS ARE THOSE INVOLVING ' ||
416:       'DEATH, BODILY INJURY, OR';
417:    CALL PRINTX (2);
418:    PRINTER = 'PROPERTY DAMAGE OF $250 OR MORE TO THE PROPERTY ' ||
419:       'OF ONE PERSON.';
420:    CALL PRINTX (1);


421:    /*** PART 1-A ***/

422:    DO I=1 TO 4;
423:       PRINTER = HDG_1A(I);
424:       IF I=1
425:          THEN CALL PRINTX (3);
426:          ELSE CALL PRINTX (1);
427:       END;
428:    PRINTER = 'NONCOLLISION';
429:    CALL PRINTX (2);
430:    OUT = ' ';
431:    T_1A = 0;
432:    DO I=1 TO 13;
433:       IF I=3
434:          THEN DO;
435:             PRINTER = 'COLLISION INVOLVING:';
436:             CALL PRINTX (1);
437:             END;
438:       O_1A.DESCR = EVENT(I);
439:       IF I=13
440:          THEN DO;
```

```
FORM16B:  PROCEDURE (STORAGE);

441:              C_1A.R2.N = T_1A(*,1);
442:              C_1A.R3.N = T_1A(*,2);
443:              END;
444:           ELSE DO;
445:              C_1A.R2.N = A_1A(I,*,1);
446:              C_1A.R3.N = A_1A(I,*,2);
447:              T_1A = T_1A + A_1A(I,*,*);
448:              END;
449:        C_1A.R2.T = C_1A.R2.N(1) + C_1A.R2.N(2) + C_1A.R2.N(3);
450:        C_1A.R3.T = C_1A.R3.N(1) + C_1A.R3.N(2) + C_1A.R3.N(3);
451:        C_1A.R1 = C_1A.R2 + C_1A.R3;
452:        O_1A.VAL = C_1A;
453:        PRINTER = OUT;
454:        CALL PRINTX (1);
455:        END;


456:     /*** PART 1-B ***/

457:     DO I=1 TO 4;
458:        PRINTER = (10)' ' || HDG_1B(I);
459:        IF I=1
460:           THEN CALL PRINTX (6);
461:           ELSE CALL PRINTX (1);
462:        END;
463:     PRINTER = (6)' ' || 'NONCOLLISION';
464:     CALL PRINTX (2);
465:     OUT = ' ';
466:     T_1B = 0;
467:     DO I=1 TO 13;
468:        IF I=3
469:           THEN DO;
470:              PRINTER = (6)' ' || 'COLLISION INVOLVING:';
471:              CALL PRINTX (1);
472:              END;
473:        O_1B.DESCR = EVENT(I);
474:        IF I=13
475:           THEN DO;
476:              C_1B.N(1) = T_1B(1);
477:              DO J=2 TO 5;
478:                 C_1B.N(J+1) = T_1B(J);
479:                 END;
480:              C_1B.N(2) = C_1B.N(3) + C_1B.N(4) + C_1B.N(5);
481:              END;
482:           ELSE DO;
483:              C_1B.N(1) = A_1B(I,1);
484:              DO J=2 TO 5;
485:                 C_1B.N(J+1) = A_1B(I,J);
486:                 END;
487:              C_1B.N(2) = C_1B.N(3) + C_1B.N(4) + C_1B.N(5);
488:              T_1B = T_1B + A_1B(I,*);
489:              END;
490:        O_1B.N = C_1B.N;
491:        PRINTER = OUT;
492:        CALL PRINTX (1);
493:        END;
```

```
FORM16B:  PROCEDURE (STORAGE);


494:     /*** PART 2-A ***/

495:     DO I=1 TO 4;
496:        PRINTER = HDG_2A(I);
497:        IF I=1
498:           THEN CALL PRINTX (9);
499:           ELSE CALL PRINTX (1);
500:        END;
501:     PRINTER = 'NONCOLLISION';
502:     CALL PRINTX (2);
503:     OUT = ' ';
504:     T_2A = 0;
505:     DO I=1 TO 13;
506:        IF I=3
507:           THEN DO;
508:              PRINTER = 'COLLISION INVOLVING:';
509:              CALL PRINTX (1);
510:              END;
511:        O_2A.DESCR = EVENT(I);
512:        IF I=13
513:           THEN DO;
514:              C_2A.R1.N = T_2A(*,1) + T_2A(*,2);
515:              C_2A.R2.N = T_2A(*,1);
516:              END;
517:           ELSE DO;
518:              C_2A.R1.N = A_2A(I,*,1) + A_2A(I,*,2);
519:              C_2A.R2.N = A_2A(I,*,1);
520:              T_2A = T_2A + A_2A(I,*,*);
521:              END;
522:        O_2A.R1 = C_2A.R1;
523:        O_2A.R2 = C_2A.R2;
524:        PRINTER = OUT;
525:        CALL PRINTX (1);
526:        END;


527:     /*** PART 3-A ***/

528:     DO I=1 TO 4;
529:        PRINTER = HDG_3(I);
530:        IF I=1
531:           THEN CALL PRINTX (6);
532:           ELSE CALL PRINTX (1);
533:        END;
534:     Z = 2;
535:     T_3A1 = 0;
536:     T_3A2 = 0;
537:     DO I=1 TO 8;
538:        OUT = ' ';
539:        O_3.DESCR = POP(I);
540:        IF I=6 | I=7 THEN GOTO PRINT_3A;
541:        IF I=8
542:           THEN DO;
543:              C_3.R2.N = T_3A1(*,1);
544:              C_3.R3.N = T_3A1(*,2);
545:              C_3.#_FAT = T_3A2(2);
```

```
FORM16B:  PROCEDURE (STORAGE);

546:              C_3.#_INJ = T_3A2(1);
547:              END;
548:           ELSE DO;
549:              C_3.R2.N = A_3A1(I,*,1);
550:              C_3.R3.N = A_3A1(I,*,2);
551:              C_3.#_FAT = A_3A2(I,2);
552:              C_3.#_INJ = A_3A2(I,1);
553:              T_3A1 = T_3A1 + A_3A1(I,*,*);
554:              T_3A2 = T_3A2 + A_3A2(I,*);
555:              END;
556:        C_3.R2.T = C_3.R2.N(1) + C_3.R2.N(2) + C_3.R2.N(3);
557:        C_3.R3.T = C_3.R3.N(1) + C_3.R3.N(2) + C_3.R3.N(3);
558:        C_3.R1 = C_3.R2 + C_3.R3;
559:        O_3.VAL = C_3;
560:  PRINT_3A:
561:        PRINTER = OUT;
562:        CALL PRINTX (Z);
563:        Z = 1;
564:        END;


565:     /*** PARTS 3-B AND 3-C ***/

566:     DO L=1 TO 2;
567:        DO I=1 TO 4;
568:           PRINTER = HDG_3(I);
569:           IF I=1
570:              THEN DO;
571:                 SUBSTR(PRINTER,1,27) = RUR_URB(L);
572:                 CALL PRINTXA (6,22);
573:                 END;
574:              ELSE DO;
575:                 SUBSTR(PRINTER,1,27) = ' ';
576:                 CALL PRINTX (1);
577:                 END;
578:           END;
579:        Z = 2;
580:        OUT = ' ';
581:        T_3A1 = 0;
582:        T_3A2 = 0;
583:        DO I=1 TO 10;
584:           IF I=10 & L=2
585:              THEN O_3.DESCR = '      TOTAL RURAL';
586:              ELSE O_3.DESCR = TRAF(I);
587:           IF I=10
588:              THEN DO;
589:                 C_3.R2.N = T_3A1(*,1);
590:                 C_3.R3.N = T_3A1(*,2);
591:                 C_3.#_FAT = T_3A2(2);
592:                 C_3.#_INJ = T_3A2(1);
593:                 END;
594:              ELSE DO;
595:                 C_3.R2.N = A_3BC1(L,I,*,1);
596:                 C_3.R3.N = A_3BC1(L,I,*,2);
597:                 C_3.#_FAT = A_3BC2(L,I,2);
598:                 C_3.#_INJ = A_3BC2(L,I,1);
599:                 T_3A1 = T_3A1 + A_3BC1(L,I,*,*);
600:                 T_3A2 = T_3A2 + A_3BC2(L,I,*);
```

```
FORM16B:  PROCEDURE (STORAGE);

601:                END;
602:           C_3.R2.T = C_3.R2.N(1) + C_3.R2.N(2) + C_3.R2.N(3);
603:           C_3.R3.T = C_3.R3.N(1) + C_3.R3.N(2) + C_3.R3.N(3);
604:           C_3.R1 = C_3.R2 + C_3.R3;
605:           O_3.VAL = C_3;
606:           PRINTER = OUT;
607:           CALL PRINTX (Z);
608:           Z = 1;
609:           END;
610:        END;


611:     /*** PART 4 ***/

612:     DO I=1 TO 3;
613:        PRINTER = HDG_4(I);
614:        IF I=1
615:           THEN CALL PRINTXA (6,23);
616:           ELSE CALL PRINTX (1);
617:        END;
618:     Z = 2;
619:     OUT = ' ';
620:     T_4 = 0;
621:     DO I=1 TO 13;
622:        O_4.DESCR = AGE(I);
623:        IF I=13
624:           THEN DO;
625:              C_4.R2.N = T_4(1,*,2);
626:              C_4.R3.N = T_4(1,*,3);
627:              C_4.R1.N = C_4.R2.N + C_4.R3.N + T_4(1,*,1);
628:              C_4.R5.N = T_4(2,*,2);
629:              C_4.R6.N = T_4(2,*,3);
630:              C_4.R4.N = T_4(2,*,1) + C_4.R5.N + C_4.R6.N;
631:              END;
632:           ELSE DO;
633:              C_4.R2.N = A_4(I,1,*,2);
634:              C_4.R3.N = A_4(I,1,*,3);
635:              C_4.R1.N = A_4(I,1,*,1) + C_4.R2.N + C_4.R3.N;
636:              C_4.R5.N = A_4(I,2,*,2);
637:              C_4.R6.N = A_4(I,2,*,3);
638:              C_4.R4.N = A_4(I,2,*,1) + C_4.R5.N + C_4.R6.N;
639:              T_4 = T_4 + A_4(I,*,*,*);
640:              END;
641:        C_4.R1.T = C_4.R1.N(1) + C_4.R1.N(2);
642:        C_4.R2.T = C_4.R2.N(1) + C_4.R2.N(2);
643:        C_4.R3.T = C_4.R3.N(1) + C_4.R3.N(2);
644:        C_4.R4.T = C_4.R4.N(1) + C_4.R4.N(2);
645:        C_4.R5.T = C_4.R5.N(1) + C_4.R5.N(2);
646:        C_4.R6.T = C_4.R6.N(1) + C_4.R6.N(2);
647:        O_4.VAL = C_4;
648:        PRINTER = OUT;
649:        CALL PRINTX (Z);
650:        Z = 1;
651:        END;


652:     /*** PART 5 ***/
```

```
FORM16B:  PROCEDURE (STORAGE);

653:     PRINTER = '5.  DIRECTION ANALYSIS.';
654:     CALL PRINTXA (6,25);
655:     PRINTER = '    AN ACCIDENT CONSISTING OF A SERIES OF ' ||
656:        'COLLISIONS, OVERTURNING, ETC., IS CLASSIFIED';
657:     CALL PRINTX (2);
658:     PRINTER = '    ACCORDING TO THE FIRST DAMAGE OR INJURY ' ||
659:        'PRODUCING EVENT; INCLUDES ON ROADWAY AND OFF ROADWAY.';
660:     CALL PRINTX (1);


661:     /*** PART 5-A ***/

662:     DO I=1 TO 3;
663:        PRINTER = (26)' ' || HDG_5A(I);
664:        IF I=1
665:           THEN CALL PRINTX (3);
666:           ELSE CALL PRINTX (1);
667:        END;
668:     OUT = ' ';
669:     Z = 2;
670:     T_5A = 0;
671:     DO I=1 TO 10;
672:        O_5A.DESCR = DIREC_5A(I);
673:        IF I=10
674:           THEN C_5A.N = T_5A;
675:           ELSE DO;
676:              C_5A.N = A_5A(I,*);
677:              T_5A = T_5A + A_5A(I,*);
678:              END;
679:        C_5A.T = C_5A.N(1) + C_5A.N(2) + C_5A.N(3);
680:        O_5A.VAL = C_5A;
681:        PRINTER = OUT;
682:        CALL PRINTX (Z);
683:        Z = 1;
684:        END;


685:     /*** PART 5-B ***/

686:     PRINTER = (26)' ' || '5B' || SUBSTR(HDG_5A(1),3);
687:     CALL PRINTXA (6,20);
688:     PRINTER = (26)' '|| HDG_5A(2);
689:     CALL PRINTX (1);
690:     PRINTER = (31)' ' || 'NOT AT INTERSECTION.' ||
691:        SUBSTR(HDG_5A(3),26);
692:     CALL PRINTX (1);
693:     T_5A = 0;
694:     OUT = ' ';
695:     Z = 2;
696:     DO I=1 TO 10;
697:        O_5A.DESCR = DIREC_5B(I);
698:        IF I=10
699:           THEN C_5A.N = T_5A;
700:           ELSE DO;
701:              C_5A.N = A_5B(I,*);
702:              T_5A = T_5A + A_5B(I,*);
703:              END;
```

```
FORM16B:  PROCEDURE (STORAGE);

704:        C_5A.T = C_5A.N(1) + C_5A.N(2) + C_5A.N(3);
705:        O_5A.VAL = C_5A;
706:        PRINTER = OUT;
707:        CALL PRINTX (Z);
708:        Z = 1;
709:        END;


710:     /*** PART 5-C ***/

711:     DO I=1 TO 3;
712:        PRINTER = (13)' ' || HDG_5C(I);
713:        IF I=1
714:           THEN CALL PRINTXA (6,16);
715:           ELSE CALL PRINTX (1);
716:        END;
717:     T_5C = 0;
718:     OUT = ' ';
719:     Z = 2;
720:     DO I=1 TO 6;
721:        O_5C.DESCR = DIREC_5C(I);
722:        IF I=6
723:           THEN DO;
724:              C_5C.R1.N = T_5C(1,*);
725:              C_5C.R2.N = T_5C(2,*);
726:              C_5C.T = T_5C(3,1) + T_5C(3,2);
727:              END;
728:           ELSE DO;
729:              C_5C.R1.N = A_5C(I,1,*);
730:              C_5C.R2.N = A_5C(I,2,*);
731:              C_5C.T = A_5C(I,3,1) + A_5C(I,3,2);
732:              T_5C = T_5C + A_5C(I,*,*);
733:              END;
734:        C_5C.R1.T = C_5C.R1.N(1) + C_5C.R1.N(2);
735:        C_5C.R2.T = C_5C.R2.N(1) + C_5C.R2.N(2);
736:        C_5C.T = C_5C.T + C_5C.R1.T + C_5C.R2.T;
737:        O_5C.VAL = C_5C;
738:        PRINTER = OUT;
739:        CALL PRINTX (Z);
740:        Z = 1;
741:        END;


742:     /*** PART 5-D ***/

743:     PRINTER = (26)' ' || '5D.  ALL OTHER ACCIDENTS.        ' ||
744:        SUBSTR(HDG_5A(1),34);
745:     CALL PRINTXA (6,28);
746:     PRINTER = (26)' ' || HDG_5A(2);
747:     CALL PRINTX (1);
748:     PRINTER = (51)' ' || SUBSTR(HDG_5A(3),26);
749:     CALL PRINTX (1);
750:     T_5A = 0;
751:     OUT = ' ';
752:     PRINTER = (20)' ' || 'AT INTERSECTION';
753:     CALL PRINTX (2);
754:     PRINTER = (23)' ' || 'COLLISION WITH:';
755:     DO I=1 TO 12;
```

```
FORM16B:  PROCEDURE (STORAGE);

756:        O_5A.DESCR = DIREC_5D(I);
757:        IF I=4 | I=9 THEN DO;
758:           PRINTER = (23)' ' || 'NONCOLLISION';
759:           CALL PRINTX (1);
760:           END;
761:        IF I=6 THEN DO;
762:           PRINTER = (20)' ' || 'NOT AT INTERSECTION';
763:           CALL PRINTX (2);
764:           PRINTER = (23)' ' || 'COLLISION WITH:';
765:           CALL PRINTX (1);
766:           END;
767:        IF I=11
768:           THEN Z = 2;
769:           ELSE Z = 1;
770:        IF I=12
771:           THEN C_5A.N = T_5A;
772:           ELSE DO;
773:              C_5A.N = A_5D(I,*);
774:              T_5A = T_5A + A_5D(I,*);
775:              END;
776:        C_5A.T = C_5A.N(1) + C_5A.N(2) + C_5A.N(3);
777:        O_5A.VAL = C_5A;
778:        PRINTER = OUT;
779:        CALL PRINTX (Z);
780:        Z = 1;
781:        END;


782:     /*** PART 6 ***/

783:     DO I=1 TO 3;
784:        PRINTER = HDG_6(I);
785:        IF I=1
786:           THEN CALL PRINTXA (6,21);
787:           ELSE CALL PRINTX (1);
788:        END;
789:     Z = 2;
790:     OUT = ' ';
791:     T_6_1 = 0;
792:     T_6_2 = 0;
793:     DO I=1 TO 12;
794:        O_6.DESCR = PED_ACT(I);
795:        IF I=12
796:           THEN DO;
797:              C_6.#_FAT = T_6_1;
798:              C_6.N = T_6_2;
799:              END;
800:           ELSE DO;
801:              C_6.#_FAT = A_6_1(I);
802:              C_6.N = A_6_2(I,*);
803:              T_6_1 = T_6_1 + A_6_1(I);
804:              T_6_2 = T_6_2 + A_6_2(I,*);
805:              END;
806:        C_6.TOT = 0;
807:        DO J=1 TO 9;
808:           C_6.TOT = C_6.TOT + C_6.N(J);
809:           END;
810:        O_6.VAL = C_6;
```

```
FORM16B:  PROCEDURE (STORAGE);

811:        PRINTER = OUT;
812:        CALL PRINTX (Z);
813:        Z = 1;
814:        END;


815:     /*** PART 7 ***/

816:     PRINTER = (81)' ' || 'ALL       FATAL      INJURY';
817:     SUBSTR(PRINTER,33,18) = '7.  AGE OF DRIVER.';
818:     CALL PRINTXA (6,22);
819:     PRINTER = (78)' ' || 'ACCIDENTS ACCIDENTS ACCIDENTS';
820:     CALL PRINTX (1);
821:     Z = 2;
822:     T_7 = 0;
823:     OUT = ' ';
824:     DO I=1 TO 13;
825:        O_7.DESCR = DRIV_AGE(I);
826:        IF I=13
827:           THEN C_7.N = T_7;
828:           ELSE DO;
829:              C_7.N = A_7(I,*);
830:              T_7 = T_7 + A_7(I,*);
831:              END;
832:        O_7.N = C_7.N;
833:        PRINTER = OUT;
834:        CALL PRINTX (Z);
835:        Z = 1;
836:        END;


837:     /*** PART 8 ***/

838:     PRINTER = (81)' ' || 'ALL       FATAL      INJURY';
839:     SUBSTR(PRINTER,33,18) = '8.  SEX OF DRIVER.';
840:     CALL PRINTXA (6,13);
841:     PRINTER = (78)' ' || 'ACCIDENTS ACCIDENTS ACCIDENTS';
842:     CALL PRINTX (1);
843:     Z = 2;
844:     T_7 = 0;
845:     OUT = ' ';
846:     DO I=1 TO 4;
847:        O_7.DESCR = SEX(I);
848:        IF I=4
849:           THEN C_7.N = T_7;
850:           ELSE DO;
851:              C_7.N = A_8(I,*);
852:              T_7 = T_7 + A_8(I,*);
853:              END;
854:        O_7.N = C_7.N;
855:        PRINTER = OUT;
856:        CALL PRINTX (Z);
857:        Z = 1;
858:        END;


859:     /*** PART 10 ***/
```

```
FORM16B:  PROCEDURE (STORAGE);

860:     PRINTER = (81)' ' || 'ALL      FATAL     INJURY';
861:     SUBSTR(PRINTER,33,17) = '10.  CONTRIBUTING';
862:     CALL PRINTXA (6,22);
863:     PRINTER = (78)' ' || 'ACCIDENTS ACCIDENTS ACCIDENTS';
864:        SUBSTR(PRINTER,38,14) = 'CIRCUMSTANCES.';
865:      CALL PRINTX (1);
866:     Z = 2;
867:     T_7 = 0;
868:     OUT = ' ';
869:     DO I=1 TO 13;
870:        O_7.DESCR = CIRC(I);
871:        IF I=13
872:           THEN C_7.N = T_7;
873:           ELSE DO;
874:              C_7.N = A_10(I,*);
875:              T_7 = T_7 + A_10(I,*);
876:              END;
877:        O_7.N = C_7.N;
878:        PRINTER = OUT;
879:        CALL PRINTX (Z);
880:        Z = 1;
881:        END;


882:     /*** PART 11 ***/

883:     PRINTER = (81)' ' || 'ALL      FATAL     INJURY';
884:     SUBSTR(PRINTER,33,21) = '11.  TYPE OF VEHICLE.';
885:     CALL PRINTXA (6,27);
886:     PRINTER = (78)' ' || 'ACCIDENTS ACCIDENTS ACCIDENTS';
887:     CALL PRINTX (1);
888:     Z = 2;
889:     T_7 = 0;
890:     OUT = ' ';
891:     DO I=1 TO 17;
892:       O_7.DESCR = VEH_TYPE(I);
893:        IF I=15 THEN DO;
894:           PRINTER = (36)' ' || 'SPECIAL VEHICLES INCLUDED ABOVE';
895:           CALL PRINTX (1);
896:           END;
897:        IF I>=15
898:           THEN C_7.N = A_11(I-1,*);
899:           ELSE IF I=14
900:              THEN C_7.N = T_7;
901:              ELSE DO;
902:                 C_7.N = A_11(I,*);
903:                 T_7 = T_7 + A_11(I,*);
904:                 END;
905:        O_7.N = C_7.N;
906:        PRINTER = OUT;
907:        CALL PRINTX (Z);
908:        Z = 1;
909:        END;


910:     /*** PART 12 ***/
```

```
FORM16B:  PROCEDURE (STORAGE);

911:     PRINTER = (81)' ' || 'ALL       FATAL      INJURY';
912:     SUBSTR(PRINTER,33,17) = '12.  ROAD SURFACE';
913:     CALL PRINTXA (6,15);
914:     PRINTER = (78)' ' || 'ACCIDENTS ACCIDENTS ACCIDENTS';
915:     SUBSTR(PRINTER,38,10) = 'CONDITION.';
916:     CALL PRINTX (1);
917:     Z = 2;
918:     T_7 = 0;
919:     OUT = ' ';
920:     DO I=1 TO 6;
921:        O_7.DESCR = ROAD(I);
922:        IF I=6
923:           THEN C_7.N = T_7;
924:           ELSE DO;
925:              C_7.N = A_12(I,*);
926:              T_7 = T_7 + A_12(I,*);
927:              END;
928:        O_7.N = C_7.N;
929:        PRINTER = OUT;
930:        CALL PRINTX (Z);
931:        Z = 1;
932:        END;


933:     /*** PART 13 ***/

934:     PRINTER = (81)' ' || 'ALL       FATAL      INJURY';
935:     SUBSTR(PRINTER,33,20) = '13.  LIGHT CONDITION';
936:     CALL PRINTXA (6,14);
937:     PRINTER = (78)' ' || 'ACCIDENTS ACCIDENTS ACCIDENTS';
938:     CALL PRINTX (1);
939:     Z = 2;
940:     T_7 = 0;
941:     OUT = ' ';
942:     DO I=1 TO 5;
943:        O_7.DESCR = LIGHT(I);
944:        IF I=5
945:           THEN C_7.N = T_7;
946:           ELSE DO;
947:              C_7.N = A_13(I,*);
948:              T_7 = T_7 + A_13(I,*);
949:              END;
950:        O_7.N = C_7.N;
951:        PRINTER = OUT;
952:        CALL PRINTX (Z);
953:        Z = 1;
954:        END;


955:     /*** PART 14 ***/

956:     PRINTER = (81)' ' || 'ALL       FATAL      INJURY';
957:     SUBSTR(PRINTER,33,24) = '14.  MANNER OF TWO MOTOR';
958:     CALL PRINTXA (6,17);
959:     PRINTER = (78)' ' || 'ACCIDENTS ACCIDENTS ACCIDENTS';
960:     SUBSTR(PRINTER,38,18) = 'VEHICLE COLLISION.';
961:     CALL PRINTX (1);
```

```
FORM16B:  PROCEDURE (STORAGE);

962:     Z = 2;
963:     T_7 = 0;
964:     OUT = ' ';
965:     DO I=1 TO 8;
966:        O_7.DESCR = TYPE(I);
967:        IF I=8
968:           THEN C_7.N = T_7;
969:           ELSE DO;
970:              C_7.N = A_14(I,*);
971:              T_7 = T_7 + A_14(I,*);
972:              END;
973:         O_7.N = C_7.N;
974:        PRINTER = OUT;
975:        CALL PRINTX (Z);
976:        Z = 1;
977:        END;


978: /***** COMPLETE THE LOOP ******/

979:     IF M2=1 THEN DO;
980:        M1, M2 = 2;
981:     PTR_STOR = ADDR(STORAGE(937));
982:     IF LOCATION='ALL'
983:        THEN SUBSTR(HEADING(3),60,15) = 'URBAN ACCIDENTS';
984:        ELSE SUBSTR(HEADING(3),52,32)='NOT LEGALLY REPORTABLE ACCIDENTS';
985:        GOTO LOOP;
986:        END;

987:     IF M2=2 THEN DO;
988:        M2 = 3;
989:        PTR_STOR = ADDR(STORAGE(1));
990:     DO I=1 TO 936;
991:        STORAGE(I) = STORAGE(I) + STORAGE(I+936);
992:           END;
993:        SUBSTR(HEADING(3),52,23) = '          ALL ACCIDENTS';
994:        GOTO LOOP;
995:        END;

996: END FORM16B;
```

CREATE-ACC-DIRECTORY --

Member Name	PNA
Language	PL/I
Subroutines	none
Files	SYSPRINT -- IBM and PNA messages ACIDENT -- Detail file ACCDIR -- Directory file (output)
Instruction	1 - 3 "PNA"

PNA scans the Detail file, and builds a directory file containing one entry for each accident in the file containing a location reference by reference post. The program checks for the presence of an I, P, or S in the first character of the location field (on system), and for a plus sign (+) in character 8 (reference post present). For the accident-by-sections listing, the number of fatalities, number of injuries, and date are required. For the multiple accident location listing, the hour, first harmful event, collision type, and road surface condition are also required. These fields are copied into the directory file to save a future reference to the detail file. The resultant file is sorted by accident number rather than by location; it must be sorted by location and accident number, and then read into an indexed-sequential file before it may be used in the report.

The PNA program listing follows:

```
/* ACCIDENT FILE DIRECTORY CREATION ROUTINE */

 1: /* ACCIDENT FILE DIRECTORY CREATION ROUTINE */

 2: GEN:  PROCEDURE OPTIONS (MAIN);

 3: /* ACCIDENT DETAIL RECORD */
 4: DECLARE
 5:    1  DET BASED (PTR_DET),
 6:       2  DUM1 CHAR(1),
 7:       2  ACC_# CHAR(12),
 8:       2  DATE CHAR(8),
 9:       2  DUM2 CHAR(26),
10:       2  MILEPOST CHAR(12),
11:       2  FIRST_EVNT CHAR(2),
12:       2  DUM3 CHAR(11),
13:       2  #_FAT_#_INJ CHAR(4),
14:       2  DUM4 CHAR(1),
15:       2  ROAD CHAR(1),
16:       2  DUM5 CHAR(14),
17:       2  TYPE CHAR(1),
18:    ACIDENT FILE INT RECORD KEYED ENV (INDEXED);

19: /* DIRECTORY FILE */
20: DECLARE
21:    OUTPUT CHAR(44) STATIC INIT (' '),
22:    1  O DEF OUTPUT POS(2),
23:       2  MILEPOST CHAR(13),
24:       2  ACC_# CHAR(12),
25:       2  #_FAT_#_INJ CHAR(4),
26:       2  DATE CHAR(8),
27:       2  FIRST_EVNT CHAR(2),
28:       2 (TYPE,ROAD) CHAR(1),
29:    ACCDIR FILE INT RECORD OUTPUT;


30: /***** INITIALIZATION *****/

31:    PUT FILE (SYSPRINT) PAGE EDIT
32:       ('ACCIDENT DIRECTORY FILE CREATION') (A);
33:    OPEN
34:       FILE (ACIDENT),
35:       FILE (ACCDIR) OUTPUT SEQL;
36:    ON ENDFILE (ACIDENT) GOTO DONE;
37:    I = 0;


38: /***** EXECUTION LOOP *****/

39: LOOP:
40:    READ FILE (ACIDENT) SET (PTR_DET);
41:    IF (SUBSTR(DET.MILEPOST,1,1)='I' | SUBSTR(DET.MILEPOST,1,1)='P' |
42:        SUBSTR(DET.MILEPOST,1,1)='S') & SUBSTR(DET.MILEPOST,8,1)='+'
43:       THEN DO;
44:          O.MILEPOST = SUBSTR(DET.MILEPOST,1,9) || '.' ||
45:             SUBSTR(DET.MILEPOST,10,3);
46:          O.ACC_# = DET.ACC_#;
47:          O.#_FAT_#_INJ = DET.#_FAT_#_INJ;
48:          O.DATE = DET.DATE;
49:          O.FIRST_EVNT = DET.FIRST_EVNT;
```

```
/* ACCIDENT FILE DIRECTORY CREATION ROUTINE */

50:           O.TYPE = DET.TYPE;
51:           O.ROAD = DET.ROAD;
52:           WRITE FILE (ACCDIR) FROM (OUTPUT);
53:           I = I + 1;
54:           END;
55:     GOTO LOOP;

56: DONE:
57:     CLOSE
58:        FILE (ACIDENT),
59:        FILE (ACCDIR);
60:     PUT FILE (SYSPRINT) SKIP(2) EDIT
61:        ('DIRECTORY FILE SUCCESSFULY CREATED') (A);
62:     PUT FILE (SYSPRINT) SKIP (2) EDIT
63:        ('NUMBER OF ENTRIES:',I) (A);

64: END GEN;
```

LOAD-ACC-DIRECTORY --

Member Name	DCA
Language	PL/I
Subroutines	none
Files	SYSPRINT -- IBM and DCA messages ACCDIR -- Sorted sequential directory ACCDIRI -- Directory file (output)
Instruction	1 - 3 "DCA"

DCA reads the sorted directory file created by CREATE-ACC-DIRECTORY, loading the records into an indexed-sequential file ACCDIRI.

The DCA program listing follows:

```
/* :LOAD-ACC-DIRECTORY */

 1: /* :LOAD-ACC-DIRECTORY */

 2: ADIRCPY:  PROCEDURE OPTIONS (MAIN);

 3: DECLARE
 4:    R CHAR(44),
 5:    ACCDIR FILE INT RECORD,
 6:    ACCDIRI FILE INT RECORD OUTPUT KEYED ENV (INDEXED);

 7:    PUT FILE (SYSPRINT) SKIP EDIT ('LOAD-ACC-DIRECTORY ROUTINE') (A);

 8:    OPEN
 9:       FILE (ACCDIR),
10:       FILE (ACCDIRI);
11:    ON ENDFILE (ACCDIR) GOTO CLOSE;
12:    I = 0;

13: LOOP:
14:    READ FILE (ACCDIR) INTO (R);
15:    WRITE FILE (ACCDIRI) FROM (R) KEYFROM (SUBSTR(R,2));
16:    I = I + 1;
17:    GOTO LOOP;

18: CLOSE:
19:    CLOSE
20:       FILE (ACCDIR),
21:       FILE (ACCDIRI);
22:    PUT FILE (SYSPRINT) SKIP(2) EDIT
23:       (I,' RECORDS IN DIRECTORY FILE') (A);

24: END ADIRCPY;
```

CREATE-ACCSUB -- CREATE-ACCSUB consists of three separate programs: CRAS for the sections phase, CRAA for the accident phase, and CRAT for the traffic phase.

PHASE=SECTIONS:

Member Name	CRAS
Language	PL/I
Subroutines	PRINTX1
Files	SYSPRINT -- IBM messages PRINTER -- CRAS output TRAFFIC -- Traffic file ROADLOG -- Roadlog file ACCSECT -- Accident report file
Instruction	1 - 4 "CRAS"

CRAS utilizes the Traffic file to generate a skeleton version of the accident report file ACCSECT. The major sections of the Traffic file are taken as sections for the accident by sections report. The description, number of lanes, and city number (municipal sections) are copied from the Roadlog file and placed into the ACCSECT file as the file is constructed.

The CRAS program listing follows:

```
CRAS: PROCEDURE(PARM) OPTIONS(MAIN);

 1: CRAS: PROCEDURE(PARM) OPTIONS(MAIN);
 2: DECLARE
 3:    ROADLOG FILE INT RECORD KEYED ENV (INDEXED),
 4:    ACCSECT FILE RECORD KEYED ENV(INDEXED),
 5:    ACC_PTR                              PTR,
 6:    CHECK_CITY                           DEC FIXED(3,0),
 7:    TRAFFIC FILE RECORD KEYED ENV(INDEXED),
 8:    TRF_PTR                              PTR,
 9:    1 TRF BASED(TRF_PTR),
10:         2 DUMMY                         CHAR(1),
11:         2 KEY                           CHAR(13),
12:         2 ROUTE_#                       DEC FIXED (3,0),
13:         2 MILEPOST                      DEC FIXED (3,0),
14:         2 FRACTION                      DEC FIXED (5,3),
15:         2 ACTUAL_ESTIMATED              CHAR(1),
16:         2 REMARK                        CHAR(1),
17:         2 DATA(4),
18:              3 YEAR                     DEC FIXED (3,0),
19:              3 ADT                      DEC FIXED (5,0),
20:              3 OUT_OF_STATE             DEC FIXED (3,3),
21:              3 PICKUPS                  DEC FIXED (3,3),
22:              3 COMMERCIAL               DEC FIXED (3,3),
23:         2 FUTURE_FACTOR                 DEC FIXED (3,3),
24:         2 DHV                           DEC FIXED (3,3),
25:         2 DATE                          CHAR(6),
26:         2 DUMMY2                        CHAR(1),
27:    TRF_CHAR                             CHAR(80),
28:    SAVE_TRF_PTR                         PTR,
29:    STRING_TRF BASED(TRF_PTR)            CHAR(80),
30:    1 ACC BASED(ACC_PTR),
31:         2 DUMMY1                        CHAR(1),
32:         2 KEY                           CHAR(13),
33:         2 REMARK                        CHAR(1),
34:         2 DESCR CHAR(35),
35:         2 SECTION_LENGTH                DEC FIXED(7,3),
36:         2 DATA(3),
37:              3 YEAR                     DEC FIXED (2,0),
38:              3 ADT                      DEC FIXED (5,0),
39:              3 #_ACC                    DEC FIXED (3,0),
40:              3 #_INJ                    DEC FIXED (3,0),
41:              3 #_FAT                    DEC FIXED (3,0),
42:              3 #_PERSONS_INJ            DEC FIXED(3,0),
43:              3 #_PERSONS_DEAD           DEC FIXED(3,0),
44:         2 #_LANES                       DEC FIXED (1,0),
45:         2 CITY_#                        DEC FIXED (3,0),
46:         2 DUMMY2                        CHAR(2),
47:    BLANK                                CHAR(120) INIT(' '),
48:    1 RLG BASED(RLG_PTR),
49:         2 DUMMY1                        CHAR(1),
50:         2 KEY                           CHAR(13),
51:         2 REMARK                        CHAR(2),
52:         2 SECTION_LENGTH                DEC FIXED(5,3),
53:         2 ROUTE_LENGTH                  DEC FIXED(5,3),
54:         2 CONSTRUCTED_LENGTH            DEC FIXED(5,3),
55:         2 UNIMPROVED_LENGTH             DEC FIXED(5,3),
56:         2 WYE_LENGTH                    DEC FIXED(3,3),
57:         2 DESCR                         CHAR(35),
58:         2 DUMMY3                        CHAR(12),
```

```
CRAS: PROCEDURE(PARM) OPTIONS(MAIN);

 59:         2 #_LANES                          DEC FIXED(1,0),
 60:         2 POPULATION                       DEC FIXED(1,0),
 61:         2 CITY_#                           DEC FIXED (3,0),
 62:     RLG_PTR                                PTR,
 63:     STRING_ACC BASED(ACC_PTR)              CHAR(104),
 64:     1 SAVE_TRF LIKE TRF BASED(SAVE_TRF_PTR),
 65:     PRINTER                                CHAR(132) EXT,
 66:     INSTR                                  CHAR(80) EXT,
 67:     PARM                                   CHAR(100),
 68:     F(0:9) STATIC                          PIC'Z'
 69:                                            INIT(0,1,2,3,4,5,6,7,8,9),
 70:     STARTKEY                               CHAR(13) DEF INSTR POS(40),
 71:     ENDKEY                                 CHAR(13) DEF INSTR POS(56);

 72: ON ENDFILE(TRAFFIC) GO TO CLOSE;

 73: CALL INIT(PARM);

 74: ACC_PTR = ADDR(BLANK);
 75: CHECK_CITY = 0;


 76: OPEN:
 77:     OPEN FILE(TRAFFIC) INPUT SEQL;
 78:     OPEN FILE(ACCSECT) OUTPUT SEQL;
 79:     OPEN FILE(ROADLOG) INPUT SEQL;


 80: START: READ FILE(TRAFFIC) SET(TRF_PTR);
 81:     IF TRF.REMARK ¬= 'T' & TRF.REMARK ¬= 'W' THEN GO TO START;
 82:     SAVE_TRF_PTR = ADDR(TRF_CHAR);


 83: LOOP:
 84:     BLANK = ((120)' ');
 85:          DO I = 1 TO 3;
 86:               ACC.DATA(I) = 0;
 87:               END;
 88:          ACC.CITY_# = 0;
 89:          ACC.SECTION_LENGTH = 0;
 90:     IF TRF.REMARK¬= 'R' & TRF.REMARK ¬= 'M' THEN DO;
 91:          TRF_CHAR = STRING_TRF;
 92:          SAVE_TRF_PTR = ADDR(TRF_CHAR);
 93:          END;
 94:     READ FILE(TRAFFIC) SET(TRF_PTR);
 95:
 96:     IF TRF.REMARK = 'R' | TRF.REMARK = 'M'
 97:     THEN GO TO LOOP;

 98: COINCIDENT:
 99:     IF TRF.REMARK='C' | TRF.REMARK='L' | TRF.REMARK='S' THEN DO;
100:     ACC.KEY = SAVE_TRF.KEY;
101:     READ FILE(ROADLOG) SET(RLG_PTR) KEY(ACC.KEY);
102:     ACC.KEY = TRF.KEY;
103:     ACC.DESCR = RLG.DESCR;
104:     ACC.REMARK = 'C';
105:     WRITE FILE(ACCSECT) FROM(STRING_ACC) KEYFROM(ACC.KEY);
106:     PRINTER = STRING_ACC;
```

```
CRAS: PROCEDURE(PARM) OPTIONS(MAIN);

107:     CALL PRINTX(F(1));
108:     READ FILE(TRAFFIC) SET(TRF_PTR);
109:     TRF_CHAR = STRING_TRF;
110:     CHECK_CITY = 0;
111:     GO TO LOOP;
112:     END;


113: END_OF_ROUTE:
114:     IF SUBSTR(SAVE_TRF.KEY,1,4) ¬= SUBSTR(TRF.KEY,1,4) THEN DO;
115:          ACC.KEY = SAVE_TRF.KEY;
116:          ACC.REMARK = 'E';
117:          READ FILE(ROADLOG) SET(RLG_PTR) KEY(ACC.KEY);
118:          ACC.DESCR = RLG.DESCR;
119:     WRITE FILE(ACCSECT) FROM(STRING_ACC) KEYFROM(ACC.KEY);
120:          PRINTER = SUBSTR(STRING_ACC,1,39);
121:          CALL PRINTX(F(1));
122:          TRF_CHAR = STRING_TRF;
123:          CHECK_CITY = 0;
124:          GO TO LOOP;
125:          END;


126: CITY:
127:     IF SAVE_TRF.REMARK = 'T' THEN DO;
128:          READ FILE(ROADLOG) SET(RLG_PTR) KEY(SAVE_TRF.KEY);
129:          IF RLG.CITY_# = 0 THEN GO TO CONTINUE;
130:          IF RLG.CITY_# = CHECK_CITY THEN GO TO LOOP;
131:          ACC.DESCR = RLG.DESCR;
132:          ACC.KEY = SAVE_TRF.KEY;
133:          ACC.REMARK = 'M';
134:          ACC.CITY_# = RLG.CITY_#;
135:          CHECK_CITY = RLG.CITY_#;
136:     WRITE FILE(ACCSECT) FROM(STRING_ACC) KEYFROM(ACC.KEY);
137:          PRINTER = SUBSTR(STRING_ACC,1,39);
138:          CALL PRINTX(F(1));
139:          TRF_CHAR = STRING_TRF;
140:          GO TO LOOP;
141:          END;

142: NON_EXISTENT:
143:     IF SAVE_TRF.REMARK='N' THEN DO;
144:          READ FILE(ROADLOG) SET(RLG_PTR) KEY(SAVE_TRF.KEY);
145:          ACC.DESCR = RLG.DESCR;
146:          ACC.KEY = SAVE_TRF.KEY;
147:          ACC.REMARK = 'N';
148:          WRITE FILE(ACCSECT) FROM(STRING_ACC) KEYFROM(ACC.KEY);
149:          PRINTER = SUBSTR(STRING_ACC,1,39);
150:          CALL PRINTX(F(1));
151:          TRF_CHAR = STRING_TRF;
152:          CHECK_CITY = 0;
153:          GO TO LOOP;
154:          END;

155: CONTINUE:
156:     ACC.KEY = SAVE_TRF.KEY;
157:     READ FILE(ROADLOG) SET(RLG_PTR) KEY(ACC.KEY);
158:     ACC.DESCR = RLG.DESCR;
```

```
CRAS: PROCEDURE(PARM) OPTIONS(MAIN);

159:     ACC.#_LANES = RLG.#_LANES;
160:          DO I = 1 TO 3;
161:          ACC.DATA(I).YEAR = SAVE_TRF.DATA(I).YEAR;
162:          END;
163:     WRITE FILE(ACCSECT) FROM(STRING_ACC) KEYFROM(ACC.KEY);
164:     PRINTER = SUBSTR(STRING_ACC,1,39);
165:     CALL PRINTX(F(1));
166:     TRF_CHAR = STRING_TRF;
167:     CHECK_CITY = 0;
168:     GO TO LOOP;


169: CLOSE:
170:     CLOSE
171:          FILE(ACCSECT),
172:          FILE(ROADLOG),
173:          FILE(TRAFFIC);
174:     CALL EXIT(PARM);
175:     END CRAS;
```

PHASE=ACCIDENT:

Member Name	CRAA
Language	PL/I
Subroutines	none
Files	SYSPRINT -- IBM and CRAA messages ACCDIRI -- Directory file ACCSECT -- Report file ACCWORK -- Scratch file ROADLOG -- Roadlog file
Instruction	1 - 4 "CRAA"

CRAA utilizes the accident directory file created and loaded by PNA and DCA to fill in the accident information of the ACCSECT file. The sections defined in the ACCSECT file (obtained from the Traffic file) are used by the program. For each section, all of the accidents occurring within the section is included in the processing unless the date of occurrence of the accident precedes the date of the road construction in the Roadlog file. After processing, each ACCSECT record contains the number of accidents occurring within the section. The multiple accident location summary requires knowledge of the number of lanes at which each accident occurs. CRAA also fills in the number of lanes into each directory file record. In addition, if the accident occurred before the construction of a new roadway (found by comparing accident date with Roadlog date), an asterisk (*) is placed into the date-flag field.

The CRAA program listing follows:

```
/* :CREATE-ACCSUB,PHASE=ACCIDENT */

 1: /* :CREATE-ACCSUB,PHASE=ACCIDENT */

 2: GEN:  PROCEDURE (PARM) OPTIONS (MAIN);

 3: /* DIRECTORY FILE */
 4: DECLARE
 5:    1  DIR STATIC,
 6:       2  DUMMY1 CHAR(1),
 7:       2  KEY CHAR(13),
 8:       2  ACC_# CHAR(12),
 9:       2 (#_FAT,#_INJ) DEC FIXED (2,0),
10:       2 (MO,DY,YR) DEC FIXED (2,0),
11:       2  DUMMY2 CHAR(6),
12:       2  #_LANES DEC FIXED (1,0),
13:       2  DATE_FLAG CHAR(1),
14:    ACCDIRI FILE INT RECORD UPDATE KEYED ENV (INDEXED);

15: /* ACCIDENT REPORT FILE */
16: DECLARE
17:   (SAVE,REP) CHAR(104),
18:    1  R BASED (PTR_REP),
19:       2  DUMMY1 CHAR(1),
20:       2  KEY CHAR(13),
21:       2  REMARK CHAR(1),
22:       2  DESCR CHAR(35),
23:       2  SECTION_LENGTH DEC FIXED (7,3),
24:       2  DATA(3),
25:          3  YR DEC FIXED (2,0),
26:          3  ADT DEC FIXED (5,0),
27:          3 (#_ACC,#_INJ_ACC,#_FAT_ACC) DEC FIXED (3,0),
28:          3 (#_INJ,#_FAT) DEC FIXED (3,0),
29:       2  #_LANES DEC FIXED (1,0),
30:       2  CITY_# DEC FIXED (3,0),
31:    1  S BASED (PTR_SAVE) LIKE R,
32:    ACCSECT FILE INT RECORD KEYED ENV (INDEXED),
33:    ACCWORK FILE INT RECORD ENV (F(1040,104));

34: /* ROADLOG FILE */
35: DECLARE
36:   (RLG,SAVE_RLG) CHAR(120),
37:    1  RD BASED (PTR_RLG),
38:       2  DUMMY1 CHAR(1),
39:       2  KEY CHAR(13),
40:       2  REMARK CHAR(2),
41:       2  DUMMY2 CHAR(61),
42:       2  #_LANES DEC FIXED (1,0),
43:       2  DUMMY3 CHAR(34),
44:       2 (YR,MO,DY) DEC FIXED (2,0),
45:    1  SRD BASED (PTR_SAVE_RLG) LIKE RD,
46:    ROADLOG FILE INT KEYED RECORD ENV (INDEXED);

47: /* OTHER VARIABLES */
48: DECLARE
49:    ENDFILE_FLAG BIN FIXED,
50:   (ACC_DATE,RLG_DATE) DEC FIXED (6,0);


51: /***** INITIALIZATION *****/
```

```
/* :CREATE-ACCSUB,PHASE=ACCIDENT */

 52:     PUT FILE (SYSPRINT) SKIP EDIT
 53:        ('     CREATE-ACCSUB ACCIDENT PHASE') (A);
 54:     PUT FILE (SYSPRINT) SKIP EDIT (' ') (A);
 55:     OPEN
 56:        FILE (ROADLOG),
 57:        FILE (ACCSECT),
 58:        FILE (ACCDIRI),
 59:        FILE (ACCWORK) OUTPUT;
 60:     ON ENDFILE (ACCSECT) ENDFILE_FLAG = 1;
 61:     ON ENDFILE (ACCDIRI) DIR.KEY = '9999';
 62:     READ FILE (ACCSECT) INTO (REP);
 63:     READ FILE (ROADLOG) INTO (RLG);
 64:     READ FILE (ACCDIRI) INTO (DIR);
 65:     PTR_REP = ADDR(REP);
 66:     PTR_SAVE = ADDR(SAVE);
 67:     PTR_RLG = ADDR(RLG);
 68:     PTR_SAVE_RLG = ADDR(SAVE_RLG);
 69:     ENDFILE_FLAG = 0;


 70: /***** EXECUTION LOOP *****/

 71: LOOP:
 72:     IF ENDFILE_FLAG¬=0 THEN GOTO COPY;
 73:     SAVE = REP;
 74:     READ FILE (ACCSECT) INTO (REP);

 75:     /* NON-BLANK REMARKS INDICATE CITIES, NON-EXISTANT SECTIONS,
 76:        COINCIDENT SECTIONS, AND ENDS OF ROUTES */
 77:     IF S.REMARK¬=' ' THEN GOTO WRITE;

 78:     /* SCAN FOR FIRST ACCIDENT IN SECTION */
 79:     DO WHILE (DIR.KEY<S.KEY);
 80:        PUT FILE (SYSPRINT) SKIP EDIT
 81:           (DIR.KEY,' DIRECTORY RECORD NOT PROCESSED -- ',
 82:            'MILEPOST DOES NOT EXIST') (A);
 83:        REWRITE FILE (ACCDIRI) FROM (DIR);
 84:        READ FILE (ACCDIRI) INTO (DIR);
 85:        END;

 86:     /* PROCESS ALL ACCIDENTS IN THE SECTION */
 87:     DO WHILE (DIR.KEY<R.KEY);

 88:        /* FIND NEAREST ROADLOG RECORD PRECEDING THE ACCIDENT
 89:           AND PLACE IN SAVE_RLG */
 90:        DO WHILE (RD.KEY<=DIR.KEY);
 91:           IF RD.REMARK='  ' | RD.REMARK='SP' | RD.REMARK='LP' |
 92:              RD.REMARK='OS' | RD.REMARK='NE'
 93:              THEN SAVE_RLG = RLG;
 94:           READ FILE (ROADLOG) INTO (RLG);
 95:           END;

 96:        /* COMPARE ROADLOG DATE TO DATE OF ACCIDENT */
 97:        ACC_DATE = 10000*DIR.YR + 100*DIR.MO + DIR.DY;
 98:        RLG_DATE = 10000*RD.YR  + 100*RD.YR  + RD.DY;
 99:        IF ACC_DATE<RLG_DATE THEN DO;
100:           PUT FILE (SYSPRINT) SKIP EDIT
```

```
/* :CREATE-ACCSUB,PHASE=ACCIDENT */

101:              (DIR.KEY,' DIRECTORY RECORD NOT PROCESSED -- ',
102:               'ROADWAY REBUILT SINCE ACCIDENT OCCURENCE')(A);
103:           DIR.DATE_FLAG = '*';
104:           GOTO NEXT_ACCIDENT;
105:           END;

106:        /* PROCESS THE ACCIDENT */
107:        DO I=1 TO 3;
108: /**TEMP**/ IF DIR.YR-1¬=S.DATA(I).YR THEN GOTO NEXT_YEAR;
109:           S.DATA(I).#_ACC = S.DATA(I).#_ACC+1;
110:           IF DIR.#_FAT¬=0
111:              THEN S.DATA(I).#_FAT_ACC = S.DATA(I).#_FAT_ACC + 1;
112:              ELSE IF DIR.#_INJ¬=0
113:                 THEN S.DATA(I).#_INJ_ACC = S.DATA(I).#_INJ_ACC + 1;
114:           S.DATA(I).#_INJ = S.DATA(I).#_INJ + DIR.#_INJ;
115:           S.DATA(I).#_FAT = S.DATA(I).#_FAT + DIR.#_FAT;
116:           GOTO NEXT_ACCIDENT;

117: NEXT_YEAR:
118:           END;
119:        PUT FILE (SYSPRINT) SKIP EDIT
120:           (DIR.KEY,' DIRECTORY RECORD NOT PROCESSED -- ',
121:            'YEAR DOES NOT MATCH COLUMN IN REPORT FILE',
122:           DIR.YR,S.DATA(1).YR,S.DATA(2).YR,S.DATA(3).YR) (A);

123: NEXT_ACCIDENT:
124:        DIR.#_LANES = SRD.#_LANES;
125:        REWRITE FILE (ACCDIRI) FROM (DIR);
126:        READ FILE (ACCDIRI) INTO (DIR);
127:        END;

128:     /* WRITE THE RECORD TO WORK FILE */

129: WRITE:
130:     WRITE FILE (ACCWORK) FROM (SAVE);
131:     PUT FILE (SYSPRINT) SKIP EDIT
132:        (S.KEY,'  ',S.REMARK,
133:        S.DATA(1).#_ACC,S.DATA(2).#_ACC,S.DATA(3).#_ACC,
134:         '   REPORT RECORD WRITTEN TO WORK FILE') (A);
135:     GOTO LOOP;


136: /***** COPY THE WORK FILE BACK TO REPORT FILE *****/

137: COPY:
138:     CLOSE
139:        FILE (ROADLOG),
140:        FILE (ACCSECT),
141:        FILE (ACCDIRI),
142:        FILE (ACCWORK);
143:     PUT FILE (SYSPRINT) SKIP (2) EDIT
144:        ('   END-OF-FILE ON ACCIDENT REPORT FILE') (A);
145:     OPEN
146:        FILE (ACCWORK),
147:        FILE (ACCSECT) OUTPUT;
148:     ON ENDFILE (ACCWORK) GOTO DONE;

149: COPY_LOOP:
```

```
/* :CREATE-ACCSUB,PHASE=ACCIDENT */

150:     READ FILE (ACCWORK) INTO (REP);
151:     WRITE FILE (ACCSECT) FROM (REP) KEYFROM (R.KEY);
152:     GOTO COPY_LOOP;

153: DONE:
154:     CLOSE
155:        FILE (ACCSECT),
156:        FILE (ACCWORK);
157:     PUT FILE (SYSPRINT) SKIP(2) EDIT
158:        ('   WORK FILE COPIED INTO ACCIDENT REPORT FILE') (A);

159: END GEN;
```

PHASE=TRAFFIC:

```
Member Name . . . . . . CRAT
Language  . . . . . . . PL/I
Subroutines . . . . . . PRINTX1
Files . . . . . . . . . SYSPRINT -- IBM messages
                        PRINTER  -- CRAT messages
                        ACCSECT  -- Accident report file
                        ACCWORK  -- Scratch file
                        TRAFFIC  -- Traffic file
                        TRUMILE  -- True Mileage file
Instruction . . . . . .  1 - 4  "CRAT"
```

PHASE=TRAFFIC calculates a weighted average daily traffic for each accident section for each of the last three consecutive years.

The CRAT program listing follows:

```
CRAT: PROCEDURE(PARM) OPTIONS(MAIN);

  1: CRAT: PROCEDURE(PARM) OPTIONS(MAIN);
  2:    DECLARE
  3:    ACCWORK FILE RECORD,
  4:    ACCSECT FILE RECORD KEYED ENV(INDEXED GENKEY),
  5:    TRAFFIC FILE RECORD KEYED ENV(INDEXED GENKEY),
  6:    TRUMILE FILE RECORD KEYED ENV(INDEXED GENKEY),
  7:    BLANK STATIC                          CHAR(104) INIT(' '),
  8:    CURRENT_ADT(3)                        PIC'ZZZZZ',
  9:    CURRENT_MILEAGE                       DEC FIXED (7,3),
 10:    CURRENT2_ADT(3)                       PIC'ZZZZZ',
 11:    ENDKEY                                CHAR(13) DEF INSTR POS(56),
 12:    F(0:9) STATIC                         PIC'Z'
 13:                                          INIT(0,1,2,3,4,5,6,7,8,9),
 14:    INSTR                                 CHAR(80) EXT,
 15:    PARM                                  CHAR(100),
 16:    PRINTER                               CHAR(120),
 17:    1 SAVE_SUBSID BASED(SAVE_SUBSID_PTR) LIKE SUBSID,
 18:    SAVE_SUBSID_PTR                       PTR,
 19:    1 SAVE_TRF LIKE TRF BASED(SAVE_TRF_PTR),
 20:    SAVE_TRF_PTR                          PTR,
 21:    STARTKEY                              CHAR(13) DEF INSTR POS(40),
 22:    STRING_SAVE_SUBSID BASED(SAVE_SUBSID_PTR) CHAR(104),
 23:    STRING_SAVE_TRF BASED(SAVE_TRF_PTR) CHAR(80),
 24:    STRING_SUBSID BASED(SUBSID_PTR) CHAR(104),
 25:    STRING_TRF BASED(TRF_PTR) CHAR(80),
 26:    STRING_TRFA BASED(TRFA_PTR) CHAR(80),
 27:    1 SUBSID BASED(SUBSID_PTR),
 28:         5 DUMMY1                         CHAR(1),
 29:         5 KEY,
 30:              10 SYSTEM                   CHAR(1),
 31:              10 ROUTE_#                  PIC'ZZZ',
 32:              10 MILEPOST                 PIC'ZZZ',
 33:              10 FRACTION                 PIC'+ZV.ZZZ',
 34:         5 REMARK                         CHAR(1),
 35:         5 DESCR                          CHAR(35),
 36:         5 SECTION_LENGTH                 DEC FIXED(7,3),
 37:         5 DATA(3),
 38:              10 YEAR                     DEC FIXED(3,0),
 39:              10 ADT                      DEC FIXED(5,0),
 40:              10 #_ACC                    DEC FIXED(3,0),
 41:              10 #_INJ                    DEC FIXED(3,0),
 42:              10 #_FAT                    DEC FIXED(3,0),
 43:              10 #_PERSON_INJ             DEC FIXED(3,0),
 44:              10 #_PERSON_DEAD            DEC FIXED(3,0),
 45:         5 #_LANES                        DEC FIXED(1,0),
 46:         5 CITY_#                         DEC FIXED(3,0),
 47:         5 DUMMY2                         CHAR(2),
 48:    SUBSID_PTR                            PTR,
 49:    1 TRF BASED(TRF_PTR),
 50:       2 DUMMY                            CHAR(1),
 51:       2 KEY                              CHAR(13),
 52:       2 ROUTE_#                          DEC FIXED (3,0),
 53:       2 MILEPOST                         DEC FIXED (3,0),
 54:       2 FRACTION                         DEC FIXED (5,3),
 55:       2 ACTUAL_ESTIMATED                 CHAR(1),
 56:       2 REMARK                           CHAR(1),
 57:       2 DATA(4),
 58:          3 YEAR                          DEC FIXED (3,0),
```

```
CRAT: PROCEDURE(PARM) OPTIONS(MAIN);

 59:          3 ADT                              DEC FIXED (5,0),
 60:          3 OUT_OF_STATE                     DEC FIXED (3,3),
 61:          3 PICKUPS                          DEC FIXED (3,3),
 62:          3 COMMERCIAL                       DEC FIXED (3,3),
 63:       2 FUTURE_FACTOR                       DEC FIXED (3,3),
 64:       2 DHV                                 DEC FIXED (3,3),
 65:       2 DATA_OF_UPDATE                      CHAR(6),
 66:       2 DUMMY2                              CHAR(2),
 67:    TRFCHAR                                  CHAR(80),
 68:    TRF_FLAG(3)                              CHAR(1),
 69:    TRF_PTR                                  PTR,
 70:    1 TRFA BASED(TRFA_PTR) LIKE TRF,
 71:    TRFA_PTR                                 PTR,
 72:    1 TRM BASED(TRM_PTR),
 73:       2 DELETE_CHAR                         CHAR(1),
 74:       2 KEY,
 75:          3 SYSTEM                           CHAR(1),
 76:          3 ROUTE_#                          PIC'999',
 77:          3 MILEPOST                         PIC'999',
 78:       2 TRUE_MILEAGE                        DEC FIXED (7,3),
 79:       2 DATE_OF_UPDATE                      DEC FIXED (7,0),
 80:    TRM_PTR                                  PTR,
 81:    TRUE_ARRAY(0:1000) STATIC                DEC FIXED (7,3),
 82:    TRUE_ROUTE                               CHAR(4),
 83:    X                                        PIC'ZZZZ9',
 84:    VEH_MILES(3)                             DEC FIXED (14,3);
 85:    ON ENDFILE(ACCSECT) GO TO COPY;
 86:    ON ENDFILE(ACCWORK) GO TO CLOSE;
 87:    OPEN FILE(ACCSECT) INPUT SEQL;
 88:    OPEN FILE(TRAFFIC) INPUT SEQL;
 89:    OPEN FILE(TRUMILE) INPUT SEQL;
 90:    OPEN FILE(ACCWORK) OUTPUT SEQL;
 91:    X = 0;
 92:    TRUE_ARRAY(0) = 0;
 93:    INSTR = PARM;
 94:    READ FILE(ACCSECT) SET(SUBSID_PTR) KEY(STARTKEY);
 95:    READ FILE(TRAFFIC) SET(TRFA_PTR) KEY(SUBSID.SYSTEM||SUBSID.ROUTE_#);
 96:    SAVE_SUBSID_PTR = ADDR(BLANK);
 97:    TRF_PTR = TRFA_PTR;
 98:    TRUE_ROUTE = '    ';
 99: LOOP: STRING_SAVE_SUBSID = STRING_SUBSID;
100:    READ FILE(ACCSECT) SET(SUBSID_PTR);
101:    IF SAVE_SUBSID.REMARK = 'M' |
102:       SAVE_SUBSID.REMARK = 'N' |
103:       SAVE_SUBSID.REMARK = 'C' |
104:         SAVE_SUBSID.REMARK = 'E' THEN DO;
105:              WRITE FILE(ACCWORK) FROM(STRING_SAVE_SUBSID);
106:          GO TO LOOP;
107:          END;
108:    IF TRUE_ROUTE ¬= (SAVE_SUBSID.SYSTEM||SAVE_SUBSID.ROUTE_#) THEN DO;
109:       TRUE_ROUTE = SAVE_SUBSID.SYSTEM||SAVE_SUBSID.ROUTE_#;
110:       READ FILE(TRUMILE) SET(TRM_PTR) KEY(TRUE_ROUTE);
111:       DO WHILE (TRM.ROUTE_# = SAVE_SUBSID.ROUTE_#);
112:          TRUE_ARRAY(TRM.MILEPOST) = TRUE_MILEAGE;
113:          READ FILE(TRUMILE) SET(TRM_PTR);
114:          END;
115:         END;
```

```
CRAT: PROCEDURE(PARM) OPTIONS(MAIN);


116:     DO WHILE (TRFA.KEY <= SUBSTR(STRING_SAVE_SUBSID,2,13));
117:     TRFCHAR = STRING_TRFA;
118:     TRF_PTR = ADDR(TRFCHAR);
119:     READ FILE(TRAFFIC) SET(TRFA_PTR);
120:     END;

121:     SAVE_TRF_PTR = TRF_PTR;
122:     DO I = 1 TO 3;
123:        CURRENT_ADT(I) = SAVE_TRF.DATA(I).ADT;
124:        END;
125:     IF (TRF.KEY ¬= SUBSTR(STRING_SAVE_SUBSID,2,13)) THEN DO;
126:        X1 = (TRUE_ARRAY(SAVE_SUBSID.MILEPOST) + SAVE_SUBSID.FRACTION)
127:        - (TRUE_ARRAY(SAVE_TRF.MILEPOST) + SAVE_TRF.FRACTION);
128:        X2 = (TRUE_ARRAY(TRFA.MILEPOST) + TRFA.FRACTION) -
129:        (TRUE_ARRAY(SAVE_SUBSID.MILEPOST) + SAVE_SUBSID.FRACTION);
130:        DO I = 1 TO 3;
131:           CURRENT_ADT(I) = SAVE_TRF.DATA(I).ADT + ((TRFA.DATA(I).ADT -
132:           SAVE_TRF.DATA(I).ADT) * (X1/(X1+X2)));
133:           END;
134:        END;
135:     VEH_MILES = 0;
136:     CURRENT_MILEAGE = (TRUE_ARRAY(SAVE_SUBSID.MILEPOST) +
137:     SAVE_SUBSID.FRACTION);
138:     IF SAVE_TRF.DATA(1).ADT = 0 THEN TRF_FLAG(1) = 'B';
139:     IF SAVE_TRF.DATA(2).ADT = 0 THEN TRF_FLAG(2) = 'B';


140:     DO WHILE ( TRFA.KEY <= SUBSTR(STRING_SUBSID,2,13));
141:        TRFCHAR = STRING_TRFA;
142:        TRF_PTR = ADDR(TRFCHAR);
143:     READ FILE(TRAFFIC) SET(TRFA_PTR);
144:        X1 = (TRUE_ARRAY(TRF.MILEPOST) + TRF.FRACTION) -
145:        CURRENT_MILEAGE;
146:        DO I = 1 TO 3;
147:           VEH_MILES(I) = VEH_MILES(I) + (CURRENT_ADT(I) +
148:           TRF.DATA(I).ADT) * (X1/2);
149:           CURRENT_ADT(I) = TRF.DATA(I).ADT;
150:           END;
151:        CURRENT_MILEAGE = (TRUE_ARRAY(TRF.MILEPOST) +
152:        TRF.FRACTION);
153:        IF TRF.DATA(1).ADT = 0 THEN TRF_FLAG(1) = 'B';
154:        IF TRF.DATA(2).ADT = 0 THEN TRF_FLAG(2) = 'B';
155:        END;


156:      IF TRF.KEY ¬= SUBSTR(STRING_SUBSID,2,13) THEN DO;
157:        X1 = (TRUE_ARRAY(SUBSID.MILEPOST) + SUBSID.FRACTION) -
158:           (TRUE_ARRAY(TRF.MILEPOST) + TRF.FRACTION);
159:        X2 = (TRUE_ARRAY(TRFA.MILEPOST) + TRFA.FRACTION) -
160:           (TRUE_ARRAY(SUBSID.MILEPOST) + SUBSID.FRACTION);
161:        DO I = 1 TO 3;
162:           CURRENT2_ADT(I) = TRF.DATA(I).ADT + (TRFA.DATA(I).ADT
163:           -  TRF.DATA(I).ADT) * (X1/(X1+X2));
164:           END;
165:        X1 = (TRUE_ARRAY(SUBSID.MILEPOST) + SUBSID.FRACTION) -
166:           CURRENT_MILEAGE;
```

```
CRAT: PROCEDURE(PARM) OPTIONS(MAIN);

167:        DO I = 1 TO 3;
168:           VEH_MILES(I) = VEH_MILES(I) + ((CURRENT_ADT(I) +
169:           CURRENT2_ADT(I)) * X1/2);
170:           END;
171:         END;
172:     SAVE_SUBSID.SECTION_LENGTH =
173:          TRUE_ARRAY(SUBSID.MILEPOST) + SUBSID.FRACTION -
174:          TRUE_ARRAY(SAVE_SUBSID.MILEPOST) + SAVE_SUBSID.FRACTION;
175:     IF TRF_FLAG(2) = 'B' THEN DO;
176:          SAVE_SUBSID.DATA(3).ADT = VEH_MILES(3)/
177:               SAVE_SUBSID.SECTION_LENGTH;
178:          SAVE_SUBSID.DATA(1).ADT = 0;
179:          SAVE_SUBSID.DATA(2).ADT = 0;
180:          END;
181:          ELSE IF TRF_FLAG(1) = 'B' THEN DO;
182:          SAVE_SUBSID.DATA(3).ADT = VEH_MILES(3)/
183:               SAVE_SUBSID.SECTION_LENGTH;
184:          SAVE_SUBSID.DATA(2).ADT = VEH_MILES(2)/
185:               SAVE_SUBSID.SECTION_LENGTH;
186:          SAVE_SUBSID.DATA(1).ADT = 0;
187:          END;
188:          ELSE DO;
189:          SAVE_SUBSID.DATA(1).ADT = VEH_MILES(1)/
190:               SAVE_SUBSID.SECTION_LENGTH;
191:          SAVE_SUBSID.DATA(2).ADT = VEH_MILES(2)/
192:               SAVE_SUBSID.SECTION_LENGTH;
193:          SAVE_SUBSID.DATA(3).ADT = VEH_MILES(3)/
194:               SAVE_SUBSID.SECTION_LENGTH;
195:          END;
196:     TRF_FLAG = ' ';

197: PRINT: WRITE FILE(ACCWORK) FROM(STRING_SAVE_SUBSID);
198:     GO TO LOOP;

199: COPY:
200:     CLOSE FILE(ACCWORK);
201:     CLOSE FILE(ACCSECT);
202:     OPEN FILE(ACCWORK)  INPUT SEQL;
203:     OPEN FILE(ACCSECT) OUTPUT SEQL;

204: COPY2: READ FILE(ACCWORK) SET(SUBSID_PTR);
205:     WRITE FILE(ACCSECT) FROM(STRING_SUBSID)
206:          KEYFROM(SUBSTR(STRING_SUBSID,2,13));
207:     X = X + 1;
208:     GO TO COPY2;

209: CLOSE:
210:     PRINTER = ' ADT HAS BEEN CALCULATED AND STORED IN  '||X||
211:          '  ACCIDENT BY SECTIONS RECORDS';
212:     PUT FILE(SYSPRINT) SKIP EDIT(PRINTER) (A);
213:     CLOSE FILE(ACCSECT);
214:     CLOSE FILE(TRAFFIC);
215:     CLOSE FILE(TRUMILE);
216:     CLOSE FILE(ACCWORK);
217:     END CRAT;
```

CREATE-ACC-LIMITS:

Member Name	CAA
Language	PL/I
Subroutines	PRINTX1
Files	SYSPRINT -- IBM messages PRINTER -- CAA messages ACCSECT -- Accident report file ACCLIM -- Limits file
Instruction	1 - 3 "CAA"

Each Accident section on the Federal Aid System is assigned an accident rate. The accident rate is determined by the number of accidents which have occurred within the section and the amount of traffic using the section. If a section of roadway has an unusually high or low accident rating for that particular Federal Aid Route the Accident By Sections Report marks that section of roadway with an asterisk. CREATE-ACC-LIMITS calculates the average accident rating for each Federal Aid Route. From this average a high and low limit is determined (see SOME COMMENTS ON THE APPLICATIONS OF STATISTICAL QUALITY CONTROL TECHNIQUES TO ACCIDENT RATES by S. K. Dietz, August, 1966) and stored in the ACCLIM File. The ACCLIM File is referred to when the Accident by Sections Report is printed.

The CAA program listing follows:

```
CAAR: PROCEDURE(PARM) OPTIONS(MAIN);

  1: CAAR: PROCEDURE(PARM) OPTIONS(MAIN);

  2: DECLARE
  3:    1 ACC BASED(ACC_PTR),
  4:         5 DUMMY1                          CHAR(1),
  5:         5 SYSTEM                         CHAR(1),
  6:         5 ROUTE_#                         PIC'ZZZ',
  7:         5 MILEPOST                       PIC'ZZZ',
  8:         5 FRACTION                       PIC'+9V.999',
  9:         5 REMARK                         CHAR(1),
 10:         5 DESCR                           CHAR(35),
 11:         5 SECTION_LENGTH                 DEC FIXED(7,3),
 12:         5 DATA(3),
 13:              10 YEAR                     DEC FIXED(2,0),
 14:              10 ADT                       DEC FIXED(5,0),
 15:              10 #_ACC                    DEC FIXED(3,0),
 16:              10 #_INJ                     DEC FIXED(3,0),
 17:              10 #_FAT                    DEC FIXED(3,0),
 18:              10 #_PERSONS_INJ            DEC FIXED(3,0),
 19:              10 #_PERSONS_DEAD            DEC FIXED(3,0),
 20:         5 #_LANES                         DEC FIXED(1,0),
 21:         5 CITY_#                         DEC FIXED(3,0),
 22:         5 DUMMY2                         CHAR(2),
 23:    ACC_PTR                                PTR,
 24:    ACCSECT FILE RECORD KEYED ENV(INDEXED),
 25:    ACCSUBR FILE RECORD KEYED ENV(INDEXED),
 26:    1 AVERAGE BASED(AVG_PTR),
 27:         5 DUMMY1                         CHAR(1),
 28:         5 KEY                            CHAR(4),
 29:         5 DATA(3),
 30:              10 LOWER_LIMIT              DEC FIXED(5,3),
 31:              10 UPPER_LIMIT              DEC FIXED(5,3),
 32:         5 DUMMY2                         CHAR(1),
 33:    AVG_ACC(3)                             DEC FIXED(9,0),
 34:    AVG_ADT(3)                             DEC FIXED(9,0),
 35:    AVG_RATE(3)                           DEC FIXED(5,3),
 36:    AVG_SECTION_LENGTH                    DEC FIXED(7,3),
 37:    AVG_PTR                               PTR,
 38:    BLANK                                 CHAR(24) INIT(' '),
 39:    COMPLETION_CODE                        CHAR(1),
 40:    F(0:9) STATIC                         PIC'Z'
 41:                                          INIT(0,1,2,3,4,5,6,7,8,9),
 42:    PARM                                  CHAR(100),
 43:    PRINTER                               CHAR(132) EXT,
 44:    SAVE_RT#                              PIC'999',
 45:    SAVE_SYSTEM                           CHAR(1),
 46:    AVG_STRING BASED(AVG_PTR)             CHAR(24);
 47:    CALL INIT(PARM);
 48:    ON ENDFILE(ACCSECT) BEGIN;
 49:         COMPLETION_CODE = 'X';
 50:         GO TO UPDATE;
 51:         END;
 52:    OPEN FILE(ACCSECT) INPUT SEQL;
 53:    OPEN FILE(ACCSUBR) OUTPUT SEQL TITLE('ACCLIM');
 54:    READ FILE(ACCSECT) SET(ACC_PTR);
 55:    AVG_PTR = ADDR(BLANK);
 56:    SAVE_RT# = ACC.ROUTE_#;
 57:    SAVE_SYSTEM = ACC.SYSTEM;
```

```
CAAR: PROCEDURE(PARM) OPTIONS(MAIN);

 58:     X = 0;
 59:     AVG_ADT = 0;
 60:     AVG_SECTION_LENGTH = 0;
 61:     AVG_ACC = 0;
 62:     COMPLETION_CODE = ' ';

 63: READ: READ FILE(ACCSECT) SET(ACC_PTR);
 64:     IF ACC.ROUTE_# ¬= SAVE_RT# THEN GO TO UPDATE;
 65: NEW_ROUTE:
 66:     IF ACC.REMARK ¬= ' ' THEN GO TO READ;
 67:     X = X + 1;
 68:     DO I = 1 TO 3;
 69:          AVG_ACC(I) = AVG_ACC(I) + ACC.DATA(I).#_ACC;
 70:          AVG_ADT(I) = AVG_ADT(I) + ACC.DATA(I).ADT;
 71:          END;
 72:          AVG_SECTION_LENGTH = AVG_SECTION_LENGTH + ACC.SECTION_LENGTH;
 73:     GO TO READ;

 74: UPDATE:
 75:     AVG_SECTION_LENGTH = AVG_SECTION_LENGTH / X;
 76:     AVERAGE.DATA = 0;
 77:     DO I = 1 TO 3;
 78:          AVG_ADT(I) = AVG_ADT(I) / X;
 79:          AVG_ACC(I) = AVG_ACC(I) / X;
 80:          AVG_RATE(I) = (AVG_ACC(I) * 1000000)/
 81:               (365 * AVG_ADT(I) * AVG_SECTION_LENGTH);
 82:          IF AVG_RATE(I) ¬= 0 THEN DO;
 83:          AVERAGE.LOWER_LIMIT(I) = AVG_RATE(I) + 1.96 * SQRT(AVG_ADT(I)
 84:               / AVG_RATE(I)) + (.48 / AVG_RATE(I))+ (1/(2 *
 85:               AVG_RATE(I)));
 86:          AVERAGE.UPPER_LIMIT = AVG_RATE(I) - 1.96 * SQRT(AVG_ADT(I)
 87:               / AVG_RATE(I)) + (.48 / AVG_RATE(I)) - (1 /(2 *
 88:               AVG_RATE(I)));
 89:           END;
 90:          END;
 91:     AVERAGE.KEY = SAVE_SYSTEM||SAVE_RT#;
 92:     WRITE FILE(ACCSUBR) FROM(AVG_STRING) KEYFROM(SAVE_SYSTEM||
 93:          SAVE_RT#);
 94:     PRINTER = AVG_STRING;
 95:     CALL PRINTX(F(1));
 96:     X = 0;
 97:     SAVE_RT# = ACC.ROUTE_#;
 98:     SAVE_SYSTEM = ACC.SYSTEM;
 99:     AVG_ADT = 0;
100:     AVG_ACC = 0;
101:     AVG_SECTION_LENGTH = 0;
102:     IF COMPLETION_CODE = 'X' THEN GO TO CLOSE;
103:     GO TO NEW_ROUTE;

104: CLOSE:
105:     CLOSE FILE(ACCSECT);
106:     CLOSE FILE(ACCSUBR);
107:     CALL EXIT(PARM);
108:     END CAAR;
109:
110:
```

ACCIDENT-BY-SECTIONS --

Member Name	ACA	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- ACA output
	ACCSECT	-- Accident report file
	ACCLIM	-- Limits file
	ROADLOG	-- Roadlog file
	CITYTBL	-- Table of city names
Instruction	1 - 3	"ACA"
	40 - 52	Beginning key
	56 - 68	Ending key

Accident-by-Sections provides a listing of the Accident Report File in a report format. The CITYTBL member of HIS.TABLES is read in order to provide the city name for each municipal accident section. The Roadlog file is accessed for descriptions of coincident sections of the roadway.

The ACA program listing follows:

```
CRAR: PROCEDURE(PARM) OPTIONS(MAIN);

  1: CRAR: PROCEDURE(PARM) OPTIONS(MAIN);

  2: DECLARE
  3:    1 ACC BASED(ACC_PTR),
  4:         5 DUMMY1                          CHAR(1),
  5:         5 SYSTEM                         CHAR(1),
  6:         5 ROUTE_#                         PIC'ZZZ',
  7:         5 MILEPOST                       PIC'ZZZ',
  8:         5 FRACTION                       PIC'+9V.999',
  9:         5 REMARK                         CHAR(1),
 10:         5 DESCR                           CHAR(35),
 11:         5 SECTION_LENGTH                 DEC FIXED(7,3),
 12:         5 DATA(3),
 13:              10 YEAR                      DEC FIXED(2,0),
 14:              10 ADT                       DEC FIXED(5,0),
 15:              10 #_ACC                     DEC FIXED(3,0),
 16:              10 #_INJ                     DEC FIXED(3,0),
 17:              10 #_FAT                     DEC FIXED(3,0),
 18:              10 #_PERSONS_INJ             DEC FIXED(3,0),
 19:              10 #_PERSONS_DEAD            DEC FIXED(3,0),
 20:         5 #_LANES                         DEC FIXED(1,0),
 21:         5 CITY_#                         DEC FIXED(3,0),
 22:         5 DUMMY2                         CHAR(2),
 23:    ACC_KEY                               CHAR(13),
 24:    ACC_PTR                               PTR,
 25:    ACCSUBR FILE RECORD KEYED ENV(INDEXED),
 26:    ACCSECT FILE INT RECORD KEYED ENV (INDEXED),
 27:    1 AVERAGE BASED(AVG_PTR),
 28:         5 DUMMY1                         CHAR(1),
 29:         5 KEY                            CHAR(4),
 30:         5 DATA(3),
 31:              10 LOWER_LIMIT              DEC FIXED(5,3),
 32:              10 UPPER_LIMIT              DEC FIXED(5,3),
 33:         5 DUMMY2                         CHAR(1),
 34:    AVG_PTR                               PTR,
 35:    CITY(126) CHAR(18),
 36:    CITY_NAME BASED(CITY_PTR)             CHAR(18),
 37:    CITY_PTR                               PTR,
 38:    ENDKEY DEF INSTR POS(56)              CHAR(13),
 39:    F(0:9) STATIC                         PIC'Z'
 40:                                          INIT(0,1,2,3,4,5,6,7,8,9),
 41:    FATAL_VALUE                           PIC'ZZZZZZ',
 42:    HEADING(9)                             CHAR(132) EXT,
 43:    INJURY_VALUE                          PIC'ZZZZZZ',
 44:    INSTR     CHAR(80) EXT,
 45:    #_CITIES                              DEC FIXED(3,0) INIT(126),
 46:    #_HDGS                                PIC'Z' DEF INSTR POS(72),
 47:    LINE(3)                               CHAR(66),
 48:    1 OUT,
 49:         5 #_LANES                         PIC'ZBBB',
 50:         5 SECTION_LENGTH                 PIC'99V.999BB',
 51:         5 ADT                             PIC'ZZZZZBB',
 52:         5 YR                              PIC'ZZBB',
 53:         5 PD                              PIC'Z9BB',
 54:         5 INJ                             PIC'Z9BB',
 55:         5 FAT                            PIC'Z9BBB',
 56:         5 TOT                             PIC'Z9BB',
 57:         5 PERSON_INJ                      PIC'Z9BBB',
```

```
CRAR: PROCEDURE(PARM) OPTIONS(MAIN);

 58:          5 PERSON_DEAD                  PIC'Z9BB',
 59:          5 PROP                          PIC'ZZZZZZBB',
 60:          5 ECON_TOT                     PIC'ZZZZZZBB',
 61:          5 RATE                         PIC'Z9V.9BB',
 62:          5 FAT_RATE                     PIC'Z9V.9BB',
 63:          5 MULTI_ACC                    PIC'Z9V.9BB',
 64:          5 MULTI_FAT                    PIC'Z9V.9BB',
 65:          5 ASTER                        CHAR(1),
 66:     PARM                                CHAR(100),
 67:     PRINTER                             CHAR(132) EXT,
 68:     PROPERTY_VALUE                      PIC'ZZZZZZ',
 69:     1 RLG BASED(RLG_PTR),
 70:          5 DUMMY1                       CHAR(1),
 71:          5 KEY                           CHAR(13),
 72:          5 DUMMY2                       CHAR(7),
 73:          5 DESCR                        CHAR(35),
 74:     RLG_PTR                             PTR,
 75:     ROADLOG FILE RECORD KEYED ENV(INDEXED),
 76:     STARTKEY DEF INSTR POS(40)          CHAR(13),
 77:     TABLE FILE RECORD,
 78:     PAGE_SIZE DEF INSTR POS(7)          PIC'ZZ',
 79:     PAGE_POSITION DEF INSTR POS(9)      PIC'ZZ';


 80: /* *** PROGRAM INITIALIZATION *** */

 81:     CALL INIT(PARM);

 82:   /* *** COLUMN HEADING *** */
 83:     #_HDGS = 5;
 84:     HEADING(3) = '  MILE     NO. SECTION                NUMBER OF ACCIDENT'
 85:          ||'  PERSONS   ECONOMIC LOSS      RATES      MULTI RATE';
 86:     HEADING(4) = '  POST   LANES LENGTH YEAR  ADT    PD INJ FAT TOTAL'
 87:          ||' INJ DEAD  PROPERTY TOTAL  TOTAL FATAL  ACCI FATAL';

 88:     /* *** READ TABLE OF CITY NAMES *** */
 89:     OPEN FILE(TABLE) INPUT RECORD TITLE('CITYTBL');
 90:     DO J = 1 TO #_CITIES;
 91:          READ FILE(TABLE) SET(CITY_PTR);
 92:          CITY(J) = CITY_NAME;
 93:          END;
 94:     CLOSE FILE(TABLE);

 95:     /* *** OPEN ALL REPORT FILES *** */
 96:     OPEN FILE(ACCSECT) INPUT SEQL;
 97:     OPEN FILE(ACCSUBR) INPUT SEQL TITLE('ACCLIM');
 98:     OPEN FILE(ROADLOG) INPUT SEQL;
 99:     ON ENDFILE(ACCSECT) GO TO CLOSE;
100:     ON KEY(ROADLOG) BEGIN;
101:          PUT FILE(SYSPRINT) SKIP EDIT(ACC_KEY) (A);
102:          END;
103:     READ FILE(ACCSUBR) SET(AVG_PTR) KEY(SUBSTR(STARTKEY,1,4));
104:     READ FILE(ACCSECT) SET(ACC_PTR) KEY(STARTKEY);
105:     IF ACC.SYSTEM = 'I' THEN HEADING(1) = (35)' ' ||
106:          'FEDERAL AID INTERSTATE ROUTE NUMBER '||ACC.ROUTE_#;
107:     IF ACC.SYSTEM = 'P' THEN HEADING(1) = (35)' '||
108:          'FEDERAL AID PRIMARY ROUTE NUMBER '|| ACC.ROUTE_#;
109:     IF ACC.SYSTEM = 'S' THEN HEADING(1) = (35)' ' ||
```

```
CRAR: PROCEDURE(PARM) OPTIONS(MAIN);

110:          'FEDERAL AID SECONDARY ROUTE NUMBER '||ACC.ROUTE_#;

111:     INJURY_VALUE = 2500;
112:     FATAL_VALUE = 41700;


113: /* *** MAIN EXECUTION LOOP *** */

114: COINCIDENT:
115:     IF ACC.REMARK = 'C' THEN DO;
116:          PRINTER = ACC.MILEPOST||ACC.FRACTION||'  '||ACC.DESCR;
117:          CALL PRINTX(F(1));
118:          IF PAGE_SIZE - PAGE_POSITION <= 5 THEN DO;
119:               PAGE_POSITION = PAGE_SIZE;
120:               CALL PRINTX(F(2));
121:               END;
122:          ACC_KEY = ACC.SYSTEM||ACC.ROUTE_#||ACC.MILEPOST||ACC.FRACTION;
123:          READ FILE(ROADLOG) SET(RLG_PTR) KEY(ACC_KEY);
124:          ACC.DESCR = RLG.DESCR;
125:          PRINTER = (29)' ' || ACC.DESCR;
126:          CALL PRINTX(F(2));
127:          PRINTER = ' ';
128:          CALL PRINTX(F(1));
129:          GO TO NEXT;
130:          END;


131: NON_EXISTENT:
132:     IF ACC.REMARK = 'N' THEN DO;
133:          PRINTER = ACC.MILEPOST||ACC.FRACTION||'  '||ACC.DESCR;
134:          CALL PRINTX(F(1));
135:          IF PAGE_SIZE - PAGE_POSITION <= 5 THEN DO;
136:               PAGE_POSITION = PAGE_SIZE;
137:               CALL PRINTX(F(2));
138:               END;
139:          PRINTER = (29)' '||'***  NON EXISTANT  ***';
140:          CALL PRINTX(F(2));
141:          PRINTER = ' ';
142:          CALL PRINTX(F(1));
143:          GO TO NEXT;
144:          END;


145: MUNICIPLE:
146:     IF ACC.REMARK = 'M' THEN DO;
147:          PRINTER = ACC.MILEPOST||ACC.FRACTION||'  '||ACC.DESCR;
148:          CALL PRINTX(F(1));
149:          IF PAGE_SIZE - PAGE_POSITION <= 5 THEN DO;
150:               PAGE_POSITION = PAGE_SIZE;
151:               CALL PRINTX(F(2));
152:               END;
153:          OUT.SECTION_LENGTH = ACC.SECTION_LENGTH;
154:          PRINTER = (15)' '||
155:               ACC.SECTION_LENGTH||(10)' '||'CITY OF '||CITY(ACC.CITY_#);
156:          CALL PRINTX(F(2));
157:          PRINTER = ' ';
158:          CALL PRINTX(F(1));
159:          GO TO NEXT;
```

```
CRAR: PROCEDURE(PARM) OPTIONS(MAIN);

160:          END;

161: END_OF_ROUTE:
162:     IF ACC.REMARK = 'E' THEN DO;
163:          PRINTER = ACC.MILEPOST||ACC.FRACTION||'  '||ACC.DESCR;
164:          CALL PRINTX(F(1));
165:          IF PAGE_SIZE - PAGE_POSITION > 2 THEN DO;
166:          PRINTER = (132)'-';
167:          CALL PRINTX(F(2));
168:          END;
169:               IF ACC.SYSTEM = 'I' THEN HEADING(1) = (35)' '||
170:               'FEDERAL AID INTERSTATE ROUTE NUMBER '||ACC.ROUTE_#;
171:               IF ACC.SYSTEM = 'P' THEN HEADING(1) = (35)' '||
172:               'FEDERAL AID ROUTE NUMBER '||ACC.ROUTE_#;
173:               IF ACC.SYSTEM = 'S' THEN HEADING(1) = (35)' '||
174:               'FEDERAL AID SECONDARY ROUTE NUMBER '||ACC.ROUTE_#;
175:               IF PAGE_SIZE - PAGE_POSITION >= 10 THEN DO;
176:               PRINTER = HEADING(1);
177:               CALL PRINTX(F(3));
178:               END;
179:          ELSE PAGE_POSITION = PAGE_SIZE;
180:     READ FILE(ACCSUBR) SET(AVG_PTR);
181:          GO TO NEXT;
182:     END;

183: CLACULATIONS:
184:     PRINTER = ACC.MILEPOST||ACC.FRACTION||'  '||ACC.DESCR;
185:     CALL PRINTX(F(2));
186:     IF PAGE_SIZE - PAGE_POSITION <= 5 THEN DO;
187:          PAGE_POSITION = PAGE_SIZE;
188:          CALL PRINTX(F(2));
189:          END;
190:     OUT.MULTI_ACC = 0;
191:     OUT.MULTI_FAT = 0;
192:     OUT.SECTION_LENGTH = ACC.SECTION_LENGTH;
193:     OUT.#_LANES = ACC.#_LANES;
194:     DO I = 1 TO 3;
195:          OUT.ADT = ACC.DATA(I).ADT;
196:          OUT.YR = ACC.DATA(I).YEAR;
197:          IF ACC.DATA(I).#_ACC = 0 THEN DO;
198:               LINE(I) = OUT.YR||OUT.ADT||(13)' '||'NONE';
199:               GO TO CONTINUE;
200:               END;
201:          OUT.PD = ACC.DATA(I).#_ACC -
202:               (ACC.DATA(I).#_INJ + ACC.DATA(I).#_FAT);
203:          OUT.INJ = ACC.DATA(I).#_INJ;
204:          OUT.FAT = ACC.DATA(I).#_FAT;
205:          OUT.TOT = ACC.DATA(I).#_ACC;
206:          OUT.PERSON_INJ = ACC.DATA(I).#_PERSONS_INJ;
207:          OUT.PERSON_DEAD = ACC.DATA(I).#_PERSONS_DEAD;
208:          OUT.PROP = PROPERTY_VALUE * ACC.DATA(I).#_ACC;
209:          OUT.ECON_TOT = OUT_PROP + ACC.DATA(I).#_INJ * INJURY_VALUE +
210:               ACC.DATA(I).#_FAT * FATAL_VALUE;
211:          OUT.RATE = (ACC.DATA(I).#_ACC * 1000000) /
212:               (365 * ACC.DATA(I).ADT * ACC.SECTION_LENGTH);
213:          OUT.FAT_RATE = (ACC.DATA(I).#_FAT * 100000000)/
214:               (365 * ACC.DATA(I).ADT * ACC.SECTION_LENGTH);
215:          OUT.MULTI_ACC = (OUT.MULTI_ACC + OUT.RATE);
```

```
CRAR: PROCEDURE(PARM) OPTIONS(MAIN);

216:          OUT.MULTI_FAT = (OUT.MULTI_FAT + OUT.FAT_RATE);
217:          IF OUT.RATE < LOWER_LIMIT(I)  | OUT.RATE > UPPER_LIMIT(I)
218:               THEN OUT.ASTER = '*';
219:          LINE(I) = OUT.YR||OUT.ADT||OUT.PD||OUT.INJ||OUT.FAT||OUT.TOT||
220:               OUT.PERSON_INJ||OUT.PERSON_DEAD||OUT.PROP||OUT.ECON_TOT||
221:               OUT.RATE||OUT.FAT_RATE;

222: CONTINUE: END;

223: PRINT:
224:     PRINTER = (11)' '||OUT.#_LANES
225:          ||OUT.SECTION_LENGTH||LINE(1)||OUT.MULTI_ACC||OUT.MULTI_FAT;
226:     CALL PRINTX(F(2));
227:     PRINTER = (23)' '||LINE(2);
228:     CALL PRINTX(F(1));
229:     PRINTER = (23)' '||LINE(3);
230:     CALL PRINTX(F(1));
231:     OUT.MULTI_ACC = 0;
232:     OUT.MULTI_FAT = 0;

233: NEXT:
234:     READ FILE(ACCSECT) SET(ACC_PTR);
235:     GO TO COINCIDENT;

236: CLOSE:
237:     CLOSE FILE(ACCSECT);
238:     CLOSE FILE(ROADLOG);
239:     CALL EXIT(PARM);
240:     END CRAR;
```

MULTIPLE-ACC-LOCNS --

Member Name	MLA	
Language	PL/I	
Subroutines	PRINTX1	
	GETDAY	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- MLA output
	ACCDIRI	-- Directory file
Instruction	1 - 3	"MLA"

MULTIPLE-ACC-LOCNS provides a listing of locations (1/10 mile sections) containing two or more accidents. This listing is similar to the one existing in previous editions of the Accident by Sections report. The "type of accident" column of the old report contains descriptions that are not exactly correspondent to codes in the present data file. This column has hence been broken down into two separate columns, one showing the "first harmful event," and the other showing the "collision type" field of the accident detail records.

The MLA program listing follows:

```
/* :MULTIPLE-LOCATION,REPORT=ACCIDENT */

 1: /* :MULTIPLE-LOCATION,REPORT=ACCIDENT */

 2: SUMMARY:  PROCEDURE (PARM) OPTIONS (MAIN);

 3: /* INSTRUCTION & PRINT ROUTINE */
 4: DECLARE
 5:    PARM CHAR(100),
 6:    INSTR CHAR(80) EXT,
 7:    #_HDGS PIC'Z' DEF INSTR POS(72),
 8:    PRINTER CHAR(132) EXT,
 9:    HEADING(9) CHAR(132) EXT,
10:    PRINTX ENTRY (PIC'Z'),
11:    PRINTXA ENTRY (PIC'Z',PIC'ZZ');

12: /* DIRECTORY FILE */
13: DECLARE
14:   (DIR,SAVE) CHAR(44) STATIC,
15:    1  D BASED (PTR_DIR),
16:       2  DUM CHAR(1),
17:       2  KEY CHAR(11),
18:       2  DUM1 CHAR(2),
19:       2  ACC_# CHAR(12),
20:       2 (#_FAT,#_INJ,MONTH,DAY,YEAR,HOUR,EVNT) DEC FIXED (2,0),
21:       2 (TYPE,SURF,#_LANES) DEC FIXED (1,0),
22:    1  S BASED (PTR_SAVE) LIKE D,
23:    ACCDIRI FILE INT RECORD KEYED ENV (INDEXED);

24: /* OUTPUT STRUCTURE */
25: DECLARE
26:    OUT CHAR(132) STATIC,
27:    1  O DEF OUT,
28:       2  RT_# CHAR(5),
29:       2  MPOST CHAR(9),
30:       2  MONTH CHAR(4),
31:       2  DAY PIC'ZZBB',
32:       2  DAY_OF_WEEK CHAR(5),
33:       2  HOUR PIC'Z9BBB',
34:       2  GROUP CHAR(7),
35:       2  #_LANES PIC'ZBBB',
36:       2  SURF CHAR(7),
37:       2  EVNT CHAR(27),
38:       2  TYPE CHAR(23);

39: /* OTHER VARIABLES */
40: DECLARE
41:    FLAG DEC FIXED (1,0),
42:    C CHAR(1),
43:   (JULIAN,DAY) DEC FIXED (3,0),
44:    MONTH(12) CHAR(3) STATIC INIT (
45:       'JAN','FEB','MAR','APR','MAY','JUN',
46:       'JUL','AUG','SEP','OCT','NOV','DEC'),
47:    DAYS(7) CHAR(3) STATIC INIT
48:       ('SAT','SUN','MON','TUE','WED','THU','FRI'),
49:    GROUP(3) CHAR(3) STATIC INIT ('FAT','INJ','PD'),
50:    SURF(0:5) CHAR(5) STATIC INIT
51:       (' ',' DRY',' WET','SNOWY',' ICY','OTHER'),
52:    EVNT(11) CHAR(25) STATIC INIT (
53:       '       OVERTURNING',
```

```
/* :MULTIPLE-LOCATION,REPORT=ACCIDENT */

 54:        '    OTHER NONCOLLISION',
 55:        '        PEDESTRIAN',
 56:        '     MV IN TRANSPORT',
 57:        '   MV IN OTHER ROADWAY',
 58:        '        PARKED MV',
 59:        '      RAILWAY TRAIN',
 60:        '       PEDALCYCLIST',
 61:        '          ANIMAL',
 62:        '       FIXED OBJECT',
 63:        '       OTHER OBJECT'),
 64:     TYPE(0:7) CHAR(20) STATIC INIT (' ',
 65:        '       HEAD ON',
 66:        '      READ END',
 67:        '        ANGLE',
 68:        ' SIDESWIPE--MEETING',
 69:        ' SIDESWIPE--PASSING',
 70:        '     BACKED INTO',
 71:        ' ');


 72: /***** INITIALIZATION *****/

 73:     ON ERROR BEGIN;
 74:        PRINTER = '*** ERROR IN MILTIPLE LOCATION ROUTINE';
 75:        CALL PRINTX (3);
 76:        GOTO QUIT;
 77:        END;

 78:     CALL INIT (PARM);
 79:     #_HDGS = 3;
 80:     HEADING(1) = 'RTE    MILE                              ACC   NO.   ROAD';
 81:     HEADING(2) = 'NO.    POST    DATE   DAY HOUR GROUP LANES  SURF' ||
 82:        '     FIRST HARMFUL EVENT       TYPE OF COLLISION';
 83:     OPEN FILE (ACCDIRI);
 84:     ON ENDFILE (ACCDIRI) D.KEY = '9';
 85:     READ FILE (ACCDIRI) INTO (SAVE);
 86:     FLAG = 0;
 87:     PTR_DIR = ADDR(DIR);
 88:     PTR_SAVE = ADDR(SAVE);


 89: /***** EXECUTION LOOP *****/

 90: LOOP:
 91:     READ FILE (ACCDIRI) INTO (DIR);
 92:     IF S.KEY=D.KEY THEN GOTO EQUAL;

 93: CONTINUE:
 94:     IF SUBSTR(D.KEY,1,1)¬=SUBSTR(S.KEY,1,1) THEN DO;
 95:        IF FLAG=0 THEN DO;
 96:           PRINTER = 'THERE WERE NO MULTIPLE ACCIDENT LOCATIONS';
 97:           CALL PRINTX (3);
 98:           END;
 99:        C = SUBSTR(D.KEY,1,1);
100:        IF C='9' THEN GOTO STOP;
101:        IF C='I' THEN PRINTER = '    INTERSTATE SYSTEM';
102:        IF C='P' THEN PRINTER = '    PRIMARY SYSTEM';
103:        IF C='S' THEN PRINTER = '    SECONDARY SYSTEM';
```

```
/* :MULTIPLE-LOCATION,REPORT=ACCIDENT */

104:        CALL PRINTXA (6,15);
105:        FLAG = 0;
106:        END;
107:     SAVE = DIR;
108:     GOTO LOOP;

109: EQUAL:      /* COME HERE WHEN A MULTIPLE LOCATION IS FOUND */
110:     FLAG = 1;
111:     OUT = ' ';
112:     O.RT_# = SUBSTR(S.KEY,2,3);
113:     O.MPOST = SUBSTR(S.KEY,5);
114:     O.MONTH = MONTH(S.MONTH);
115:     O.DAY = S.DAY;
116:     CALL GETDAY (S.MONTH,S.DAY,S.YEAR,JULIAN,DAY);
117:     O.DAY_OF_WEEK = DAYS(DAY);
118:     O.HOUR = S.HOUR;
119:     IF S.#_FAT¬=0
120:        THEN I = 1;
121:        ELSE IF S.#_INJ¬=0
122:           THEN I = 2;
123:           ELSE I = 3;
124:     O.GROUP = GROUP(I);
125:     O.#_LANES = S.#_LANES;
126:     O.SURF = SURF(S.SURF);
127:     O.EVNT = EVNT(S.EVNT);
128:     O.TYPE = TYPE(S.TYPE);
129:     PRINTER = OUT;
130:     CALL PRINTXA (2,4);
131:     DO WHILE (D.KEY=S.KEY);
132:        OUT = ' ';
133:        O.MONTH = MONTH(D.MONTH);
134:        O.DAY = D.DAY;
135:        CALL GETDAY (D.MONTH,D.DAY,D.YEAR,JULIAN,DAY);
136:        O.DAY_OF_WEEK = DAYS(DAY);
137:        O.HOUR = D.HOUR;
138:        IF D.#_FAT¬=0
139:           THEN I = 1;
140:           ELSE IF D.#_INJ¬=0
141:              THEN I = 2;
142:              ELSE I = 3;
143:        O.GROUP = GROUP(I);
144:        O.#_LANES = D.#_LANES;
145:        O.SURF = SURF(D.SURF);
146:        O.EVNT = EVNT(D.EVNT);
147:        O.TYPE = TYPE(D.TYPE);
148:        PRINTER = OUT;
149:        CALL PRINTX (1);
150:        READ FILE (ACCDIRI) INTO (DIR);
151:        END;
152:     GOTO CONTINUE;

153: STOP:
154:     CLOSE FILE (ACCDIRI);
155:     CALL EXIT (PARM);

156: QUIT:

157: END SUMMARY;
```

CHAPTER 2-V
SUFFICIENCY PROGRAMMER INFORMATION

Introduction

This chapter presents a description of the programs comprising the Sufficiency Subsystem of HIS (Highway Information System). It is designed for utilization with the publication Highway Information System Volume 1: User Information.

Sufficiency File Description

Data Set Name	HIS.SUFFICY
Organization	Indexed Sequential
Logical Record Length	64
Physical Record Length	640
Key Length	13
Volume Serial Number	231428

The format of a Sufficiency record is shown in PL/I terminology in Figure 2-V-1.

Sufficiency Report File Description

Data Set Name	HIS.SUFFSUB
Organization	Indexed Sequential
Logical Record Length	120
Physical Record Length	1200
Key Length	13
Volume Serial Number	231428

The Sufficiency Report file record structure is shown in PL/I terminology in Figure 2-V-2. The data items in the file are derived from the Roadlog, Traffic, Traffic Summary, True Mileage, Sufficiency, and Accident Directory file.

```
1   SUFFICIENCY_RECORD,
    2  DELETE_CHARACTER                  CHAR(1),
    2  KEY,
       3  ROUTE_SYSTEM                   CHAR(1),
       3  ROUTE_NUMBER                   CHAR(3),
       3  REFERENCE_POST                 CHAR(3),
       3  DISTANCE                       CHAR(6),
    2  DESCRIPTION                       CHAR(18),
    2  DESIGN_SPEED                      PIC'ZZ',
    2  TERRAIN                           PIC'Z',
    2  AVERAGE_SPEED                     PIC'ZZ',
    2  SIGHT_DISTANCE                    PIC'ZZ',
    2  STOPPING_DISTANCE                 PIC'ZZ',
    2  NUMBER_OF_CURVES                  PIC'ZZ',
    2  NUMBER_OF_NARROW_BRIDGES          PIC'Z',
    2  FOUNDATION_RATING                 PIC'ZZ',
    2  SURFACE_RATING                    PIC'ZZ',
    2  DRAINAGE_RATING                   PIC'ZZ',
    2  SECTION_LENGTH                    PIC'ZZZVZZZ',
    2  DATE_OF_UPDATE,
       3  YEAR                           CHAR(2),
       3  MONTH                          CHAR(2),
       3  YEAR                           CHAR(2),
    2  DUMMY                             CHAR(2);
```

Figure 2-V-1. Sufficiency file structure.

```
1   SUFFICIENCY_REPORT_RECORD,
    2  DELETE_CHARACTER                  CHAR(1),
    2  KEY,
       3  ROUTE_SYSTEM                   CHAR(1),
       3  ROUTE_NUMBER                   CHAR(3),
       3  REFERENCE_POST                 CHAR(3),
       3  DISTANCE                       CHAR(6),
    2  REMARK                            CHAR(1),
    2  DESCRIPTION                       CHAR(18),
    2  COUNTY_NUMBER                     PIC'ZZ',
    2  FINANCIAL_DISTRICT                PIC'ZZ',
    2  YEAR_BUILT                        PIC'ZZ',
    2  YEAR_IMPROVED                     PIC'ZZ',
    2  SURFACE_WIDTH                     PIC'ZZ',
    2  ROADWAY_WIDTH                     PIC'ZZ',
    2  SURFACE_TYPE                      CHAR(3),
    2  SECTION_LENGTH                    PIC'ZZZVZZZ',
    2  AVERAGE_DAILY_TRAFFIC             PIC'ZZZZZ',
    2  DESIGN_HOUR_VOLUME                PIC'ZZZZ',
    2  PERCENT_COMMERCIAL                PIC'ZZ',
    2  SERVICE_VOLUME                    PIC'ZZZZ',
    2  NUMBER_OF_ACCIDENTS               PIC'ZZ',
    2  FOUNDATION_RATING                 PIC'ZZ',
    2  SURFACE_RATING                    PIC'ZZ',
    2  DRAINAGE_RATING                   PIC'ZZ',
    2  SAFETY_RATING                     PIC'ZZ',
    2  CAPACITY_RATING                   PIC'ZZ',
    2  TOTAL_RATING                      PIC'ZZ',
    2  ADJUSTED_RATING                   PIC'ZZ',
    2  DEFICIENT_MILEAGE                 PIC'ZZVZZ',
    2  DESIGN_SPEED                      PIC'ZZ',
    2  TERRAIN                           PIC'Z',
    2  AVERAGE_SPEED                     PIC'ZZ',
    2  SIGHT_DISTANCE                    PIC'ZZ',
    2  STOPPING_DISTANCE                 PIC'ZZ',
    2  NUMBER_OF_CURVES                  PIC'ZZ',
    2  NUMBER_OF_NARROW_BRIDGES          PIC'Z',
    2  NUMBER_OF_LANES                   PIC'Z',
    2  DIVIDED_UNDIVIDED_CODE            CHAR(1),
    2  CITY_NUMBER                       PIC'ZZZ',
    2  CURRENT_AVERAGE_DAILY_TRAFFIC     PIC'ZZZZZ',
    2  DUMMY                             CHAR(9);
```

Figure 2-V-2. Sufficiency report file structure.

SRTYPR Subroutine

The SRTYPR subroutine, which converts the 4-digit Roadlog surface type into a simplified 1-digit code, is utilized within the Sufficiency subsystem. This subroutine is described above in Chapter 2-II.

Program Descriptions

Each program in the Sufficiency Subsystem is stored in load module format in cataloged library HIS.LOADLIB, from which it is retrieved for execution by the HIS supervisor when requested. The member name for each program is given with the program description.

This section of the manual presents a write-up on each program in the Sufficiency Subsystem. An attempt has been made in the source listing itself to document the programs with appropriate variable names and comments.

CREATE-SUFFSUB -- CREATE-SUFFSUB is comprised of five separate programs. These programs store data from the Sufficiency, Roadlog, Traffic, and Accident Data Files, and calculate all of the necessary sufficiency ratings.

PHASE=SUFFICIENCY:

```
Member Name . . . . . . . SYSS
Language . . . . . . . . PL/I
Subroutine . . . . . . . PRINTX1
Files . . . . . . . . . . SYSPRINT -- IBM messages
                          PRINTER  -- "dump" listing
                          SUFFICY  -- Sufficiency file
                          SUFFSUB  -- Sufficiency Report file
                          TRUMILE  -- True Mileage file
Instruction . . . . . .  1 - 4   "SYSS"
```

This program retrieves data from the Sufficiency file and stores the data in the Sufficiency Report file. SYSS must be run before any other CREATE-SUFFSUB phase. SYSS also utilizes the True Mileage file and the milepoints retrieved from the Sufficiency file to calculate a sufficiency section length. The sufficiency section length is then stored in the Sufficiency Report file.

The SYSS program listing follows:

```
SXSS: PROCEDURE(PARM) OPTIONS(MAIN);

 1: SXSS: PROCEDURE(PARM) OPTIONS(MAIN);
 2: /* PROGRAM TO LOAD DATA FROM THE SUFFICIENCY DATA
 3:    FILE INTO THE SUFFICIENCY SUBSIDIARY FILE */


 4: /* FILE DECLARATION */

 5: DECLARE SUFFICY FILE RECORD KEYED ENV(INDEXED);
 6: DECLARE SUFFSUB FILE RECORD KEYED ENV(INDEXED);
 7: DECLARE TRUMILE FILE RECORD KEYED ENV(INDEXED GENKEY);


 8: /* VARIABLE DECLARATIONS */

 9:    DECLARE
10:    HEADING(9)                          CHAR(132) EXT,
11:    INSTR                               CHAR(80) EXT,
12:    PARM                                CHAR(100),
13:    PRINTER                             CHAR(132) EXT,
14:    PRINTX ENTRY (PIC'Z'),
15:    SAVE_FRACTION PIC'+9V.999',
16:    SAVE_MILEPOST PIC'ZZZ',
17:    STRING_SUF BASED(SUF_PTR) CHAR(64),
18:    1 SUF BASED(SUF_PTR),
19:         5 DUMMY1                       CHAR(1),
20:         5 SYSTEM                       CHAR(1),
21:         5 ROUTE_#                      PIC'ZZZ',
22:         5 MILEPOST                     PIC'ZZZ',
23:         5 FRACTION                     PIC'+9V.999',
24:         5 DESCR                        CHAR(18),
25:         5 DESIGN_SPEED                 PIC'ZZ',
26:         5 TERRAIN                      PIC'Z',
27:         5 AVG_SPEED                    PIC'ZZ',
28:         5 SIGHT_DIST                   PIC'ZZ',
29:         5 STOP_DIST                    PIC'ZZ',
30:         5 CURVES                       PIC'ZZ',
31:         5 BRIDGES                      PIC'Z',
32:         5 FOUNDATION                   PIC'ZZ',
33:         5 SURFACE                      PIC'ZZ',
34:         5 DRAINAGE                     PIC'ZZ',
35:         5 TEMP_LENGTH                  PIC'ZZZVZZZ',
36:         5 DATE                         CHAR(6),
37:     STRING_SUBSID CHAR(120) STATIC,
38:    SUF_PTR                             PTR,
39:     1  SUBSID DEF STRING_SUBSID,
40:         5 DUMMY1                       CHAR(1),
41:         5 SYSTEM                       CHAR(1),
42:         5 ROUTE_#                      PIC'ZZZ',
43:         5 MILEPOST                     PIC'ZZZ',
44:         5 FRACTION                     PIC'+9V.999',
45:         5 REMARK                       CHAR(1),
46:         5 DESCR                        CHAR(18),
47:         5 COUNTY_#                     PIC'ZZ',
48:         5 FINANCIAL_DISTRICT           PIC'ZZ',
49:         5 YR_BLT                       PIC'ZZ',
50:         5 YR_IMP                       PIC'ZZ',
51:         5 SUR_WD                       PIC'ZZ',
52:         5 RDY_WD                       PIC'ZZ',
```

```
SXSS: PROCEDURE(PARM) OPTIONS(MAIN);

 53:          5 SURTYP                          PIC'ZZZ',
 54:          5 SECTION_LENGTH                  PIC'ZZZVZZZ',
 55:          5 ADT                             PIC'ZZZZZ',
 56:          5 DHV                             PIC'ZZZZ',
 57:          5 PERCENT_TRUCKS                  PIC'ZZ',
 58:          5 SERVICE_VOL                     PIC'ZZZZ',
 59:          5 #_ACCIDENTS                     PIC'ZZ',
 60:          5 FOUNDATION                      PIC'ZZ',
 61:          5 SURFACE                         PIC'ZZ',
 62:          5 DRAINAGE                        PIC'ZZ',
 63:          5 SAFTEY_RATING                   PIC'ZZ',
 64:          5 CAPACITY_RATING                 PIC'ZZ',
 65:          5 TOTAL_RATING                    PIC'ZZZ',
 66:          5 ADJ_RATING                      PIC'ZZZ',
 67:          5 DEFICIENT_MILEAGE               PIC'ZZVZZ',
 68:          5 DESIGN_SPEED                    PIC'ZZ',
 69:          5 TERRAIN                         PIC'Z',
 70:          5 AVG_SPEED                       PIC'ZZ',
 71:          5 SIGHT_DIST                      PIC'ZZ',
 72:          5 STOP_DIST                       PIC'ZZ',
 73:          5 CURVES                          PIC'ZZ',
 74:          5 BRIDGES                         PIC'Z',
 75:          5 #_LANES                         PIC'Z',
 76:          5 DIVIDED_CODE                    CHAR(1),
 77:          5 CITY_#                          PIC'ZZZ',
 78:          5 CURRENT_SECTION_ADT             PIC'ZZZZZ',
 79:     1 TRM BASED(TRM_PTR),
 80:          5 DUMMY1                          CHAR(1),
 81:          5 SYSTEM                          CHAR(1),
 82:          5 ROUTE_#                         PIC'ZZZ',
 83:          5 MILEPOST                        PIC'ZZZ',
 84:          5 TRUE_MILEAGE                    DEC FIXED(7,3),
 85:          5 DUMMY2                          DEC FIXED(7,0),
 86:     TRM_PTR                                PTR,
 87:     TRUE_ARRAY(0:999) DEC FIXED(7,3) STATIC,
 88:     TRUE_ROUTE                             CHAR(4),
 89:     TRUE_KEY                               CHAR(7);


 90:     CALL INIT(PARM);


 91:     ON ENDFILE(SUFFICY) GO TO CLOSE;


 92: /* OPEN FILES */

 93:     OPEN FILE(SUFFICY) INPUT SEQL;
 94:     OPEN FILE(SUFFSUB) OUTPUT SEQL;
 95:     OPEN FILE(TRUMILE) INPUT SEQL;

 96:     TRUE_ROUTE = '    ';


 97: /* READ THE SUFFICIENCY FILE */

 98:  BEGIN: READ FILE(SUFFICY) SET(SUF_PTR);
```

```
SXSS: PROCEDURE(PARM) OPTIONS(MAIN);


 99: /* ALLOCATE INPUT SUFFICIENCY DATA TO THE OUTPUT SUBFILE STRUCTURE */


100: ALLOCATE: STRING_SUBSID = ' ';
101:    SUBSID = SUF,BY NAME;


102: /* CALCULATE THE SECTION LENGTH FOR EACH SUFFICIENCY SECTION */
103:    IF TRUE_ROUTE = SUBSTR(STRING_SUF,2,4) THEN GO TO LENGTH;
104:    TRUE_ROUTE = SUBSTR(STRING_SUF,2,4);
105:    READ FILE(TRUMILE) SET(TRM_PTR) KEY(TRUE_ROUTE);
106:    DO WHILE (TRM.ROUTE_# = SUF.ROUTE_#);
107:         TRUE_ARRAY(TRM.MILEPOST) = TRM.TRUE_MILEAGE;
108:         READ FILE(TRUMILE) SET(TRM_PTR);
109:         END;

110: LENGTH: SAVE_MILEPOST = SUF.MILEPOST;
111:    SAVE_FRACTION = SUF.FRACTION;
112:    READ FILE(SUFFICY) SET(SUF_PTR);
113:    IF SUBSID.DESCR = 'COINCIDENT          ' THEN GO TO CHECK;
114:    IF SUBSID.DESCR = 'END OF ROUTE       ' THEN GO TO PRINT;
115:    IF SUBSID.DESIGN_SPEED = 0 | SUBSID.DESIGN_SPEED = 1 THEN DO;
116:         SUBSID.SECTION_LENGTH = TEMP_LENGTH;
117:         IF SUBSID.DESIGN_SPEED = 0 THEN
118:              SUBSID.DEFICIENT_MILEAGE = TEMP_LENGTH;
119:         GO TO CHECK;
120:         END;
121:    SUBSID.SECTION_LENGTH = (TRUE_ARRAY(SUF.MILEPOST) + SUF.FRACTION)
122:    - (TRUE_ARRAY(SAVE_MILEPOST) + SAVE_FRACTION);


123: /* CHECK FOR UNDER CONSTRUCTION, NON EXISTANT, CITY,
124:    & COINCIDENT MILEAGE */

125: CHECK: IF SUBSID.DESIGN_SPEED = 0 THEN SUBSID.REMARK = 'N';
126:    IF SUBSID.DESIGN_SPEED = 1 THEN SUBSID.REMARK = 'U';
127:    IF SUBSID.DESCR = 'CITY                ' THEN SUBSID.REMARK= 'M';
128:    IF SUBSID.DESCR = 'COINCIDENT          ' THEN SUBSID.REMARK= 'C';


129: /* CREATE THE SUBSIDIARY FILE */
130: PRINT: WRITE FILE(SUFFSUB) FROM(STRING_SUBSID)
131:    KEYFROM(SUBSTR(STRING_SUBSID,2,13));
132: PRINTER = STRING_SUBSID;
133:     CALL PRINTX(1);
134:    GO TO ALLOCATE;


135: /* CLOSE FILES */
136: CLOSE: WRITE FILE(SUFFSUB) FROM(STRING_SUBSID)
137:    KEYFROM(SUBSTR(STRING_SUBSID,2,13));
138:    CLOSE FILE(SUFFSUB);
139:    CLOSE FILE(SUFFICY);
140:    CLOSE FILE(TRUMILE);
141:    CALL EXIT(PARM);
142:    END SXSS;
```

PHASE=ROADLOG:

Member Name	SYSR	
Language	PL/I	
Subroutine	PRINTX1	
	SRTYPR	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- "dump" listing
	ROADLOG	-- Roadlog file
	SUFFSUB	-- Sufficiency Report file
Instruction	1 - 4	"SYSR"

This program retrieves the county number, year built, year improved, surface width, roadway width, surface type, divided highway code, number of lanes, and city number from the Roadlog Data File, and stores this data in the Sufficiency Report file.

The SYSR program listing follows:

```
SYSR: PROCEDURE(PARM) OPTIONS(MAIN);

  1: SYSR: PROCEDURE(PARM) OPTIONS(MAIN);


  2: /* FILE DECLARATIONS */

  3: DECLARE ROADLOG FILE RECORD KEYED ENV(INDEXED);
  4: DECLARE SUFFSUB FILE RECORD KEYED ENV(INDEXED);
  5: DECLARE TABLE FILE RECORD;


  6: /* VARIABLE DECLARATIONS */
  7:    DECLARE
  8:    CHAR80                                CHAR(80),
  9:    ENDKEY                                CHAR(13) DEF INSTR POS(56),
 10:    F(0:9)                                PIC'Z'
 11:                                          INIT(0,1,2,3,4,5,6,7,8,9),
 12:    FINAN_DIST(0:56)                      PIC'ZZ',
 13:    INSTR                                 CHAR(80) EXT,
 14:    PARM                                  CHAR(100),
 15:    PRINTER                               CHAR(132) EXT,
 16:    1 RLG BASED(RLG_PTR),
 17:         5 DUMMY1                         CHAR(1),
 18:         5 KEY,
 19:              10 SYSTEM                   PIC'Z',
 20:              10 ROUTE_#                  PIC'ZZZ',
 21:              10 MILEPOST                 PIC'ZZZ',
 22:              10 FRACTION                 PIC'+ZV.ZZZ',
 23:         5 REMARK                         CHAR(2),
 24:         5 DUMMY2                         CHAR(14),
 25:         5 DESCRIPTION                    CHAR(35),
 26:         5 DUMMY3                         CHAR(11),
 27:         5 DIVIDED_CODE                   CHAR(1),
 28:         5 #_LANES                        DEC FIXED (1,0),
 29:         5 POP_CODE                       DEC FIXED (1,0),
 30:         5 CITY_#                         DEC FIXED (3,0),
 31:         5 COUNTY_#                       DEC FIXED (2,0),
 32:         5 YR_BLT                         DEC FIXED (2,0),
 33:         5 YR_IMP                         DEC FIXED (2,0),
 34:         5 DUMMY4                         CHAR(8),
 35:         5 SUR_WD                         DEC FIXED (2,0),
 36:         5 RDY_WD                         DEC FIXED (2,0),
 37:         5 DUMMY5                         CHAR(5),
 38:         5 SURTYP                         DEC FIXED (4,0),
 39:    RLG_KEY                               CHAR(13),
 40:    1 SAVE_RLG BASED(SAVE_RLG_PTR) LIKE RLG,
 41:    SRTYP(0:8)                            CHAR(3)
 42:                                          INIT('***','PRM','BLD','GRD',
 43:                                          'GRV','BST','RMS','PMS','PCC'),
 44:    STARTKEY                              CHAR(13) DEF INSTR POS(40),
 45:    STRING_RLG BASED(RLG_PTR) CHAR(136),
 46:    STRING_SUBSID BASED(SUB_PTR) CHAR(120),
 47:    1 SUBSID BASED(SUB_PTR),
 48:         5 DUMMY1                         CHAR(1),
 49:         5 KEY,
 50:              10 SYSTEM                   PIC'Z',
 51:              10 ROUTE_#                  PIC'ZZZ',
 52:              10 MILEPOST                 PIC'ZZZ',
 53:              10 FRACTION                 PIC'+9V.999',
```

```
SYSR: PROCEDURE(PARM) OPTIONS(MAIN);

 54:          5 REMARK                        CHAR(1),
 55:          5 DESCR                         CHAR(18),
 56:          5 COUNTY_#                      PIC'ZZ',
 57:          5 FINANCIAL_DISTRICT            PIC'ZZ',
 58:          5 YR_BLT                        PIC'ZZ',
 59:          5 YR_IMP                        PIC'ZZ',
 60:          5 SUR_WD                        PIC'ZZ',
 61:          5 RDY_WD                        PIC'ZZ',
 62:          5 SURTYP                        CHAR(3),
 63:          5 SECTION_LENGTH                PIC'ZZZVZZZ',
 64:          5 ADT                           PIC'ZZZZZ',
 65:          5 DHV                           PIC'ZZZZ',
 66:          5 PERCENT_TRUCKS                PIC'ZZ',
 67:          5 SERVICE_VOL                   PIC'ZZZZ',
 68:          5 #_ACCIDENTS                   PIC'ZZ',
 69:          5 FOUNDATION                    PIC'ZZ',
 70:          5 SURFACE                       PIC'ZZ',
 71:          5 DRAINAGE                      PIC'ZZ',
 72:          5 SAFTEY_RATING                 PIC'ZZ',
 73:          5 CAPACITY_RATING               PIC'ZZ',
 74:          5 TOTAL_RATING                  PIC'ZZZ',
 75:          5 ADJ_RATING                    PIC'ZZZ',
 76:          5 DEFICIENT_MILEAGE             PIC'ZZVZZ',
 77:          5 DESIGN_SPEED                  PIC'ZZ',
 78:          5 TERRAIN                       PIC'Z',
 79:          5 AVG_SPEED                     PIC'ZZ',
 80:          5 SIGHT_DIST                    PIC'ZZ',
 81:          5 STOP_DIST                     PIC'VZZ',
 82:          5 CURVES                        PIC'ZZ',
 83:          5 BRIDGES                       PIC'Z',
 84:          5 #_LANES                       PIC'Z',
 85:          5 DIVIDED_CODE                  CHAR(1),
 86:          5 CITY_#                        PIC'ZZZ',
 87:          5 CURRENT_SECTION_ADT           PIC'ZZZZZ',
 88:     SUB_PTR                              PTR,
 89:     SUBSID_KEY                           CHAR(13);


 90: /* OPEN FILES */

 91:     OPEN FILE(ROADLOG) INPUT SEQL;
 92:     OPEN FILE(SUFFSUB) UPDATE SEQL;


 93:     CALL INIT(PARM);
 94:     CALL SRTYPRI;


 95:     ON ENDFILE(SUFFSUB) GO TO CLOSE;


 96: /*  INITIALIZE TABLE OF FINANCIAL DISTRICTS */

 97:     OPEN FILE(TABLE) INPUT SEQL TITLE('CNTYTBL');
 98:     DO I = 1 TO 56;
 99:          READ FILE(TABLE) INTO(CHAR80);
100:          FINAN_DIST(I) = SUBSTR(CHAR80,50,2);
101:          END;
```

```
SYSR: PROCEDURE(PARM) OPTIONS(MAIN);

102:    CLOSE FILE(TABLE);
103:    FINAN_DIST(0) = 0;
104:    READ FILE(SUFFSUB) SET(SUB_PTR) KEY(STARTKEY);
105:    SUBSID_KEY = SUBSTR(STRING_SUBSID,2,13);
106:    READ FILE(ROADLOG) SET(RLG_PTR) KEY(SUBSID.SYSTEM||
107:         SUBSID.ROUTE_#||'          ');
108:    SAVE_RLG_PTR = RLG_PTR;
109:    RLG_KEY = SUBSTR(STRING_RLG,2,13);


110: /* CHECK THE SUFFICIENCY REMARK TO DETERMINE
111:    WHAT ROADLOG INFORMATION IS NECESSARY */

112: CHECK: IF SUBSID.DESCR = 'END OF ROUTE       ' THEN GO TO REWRITE;
113:    IF SUBSID.REMARK = 'C' THEN GO TO REWRITE;


114: /* LOCATE THE NECESSARY ROADLOG INFORMATION */
115:    DO WHILE (RLG_KEY <= SUBSID_KEY);
116:         IF RLG.REMARK = '  ' THEN SAVE_RLG_PTR = RLG_PTR;
117:         READ FILE(ROADLOG) SET(RLG_PTR);
118:         RLG_KEY = SUBSTR(STRING_RLG,2,13);
119:         END;


120: /* UPDATE THE SUBSIDIARY FILE WITH THE NECESSARY ROADLOG INFORMATION*/

121:    IF RLG_KEY = SUBSID_KEY THEN SAVE_RLG_PTR = RLG_PTR;
122:    SUBSID.COUNTY_# = SAVE_RLG.COUNTY_#;
123:    SUBSID.FINANCIAL_DISTRICT = FINAN_DIST(SAVE_RLG.COUNTY_#);
124:    IF SUBSID.REMARK = 'N' | SUBSID.REMARK = 'U' THEN GO TO REWRITE;
125:    SUBSID.CITY_# = SAVE_RLG.CITY_#;
126:    IF SUBSID.REMARK = 'M' THEN GO TO REWRITE;
127:    SUBSID.#_LANES = SAVE_RLG.#_LANES;
128:    SUBSID.DIVIDED_CODE = SAVE_RLG.DIVIDED_CODE;
129:    SUBSID.YR_BLT = SAVE_RLG.YR_BLT;
130:    SUBSID.RDY_WD = SAVE_RLG.RDY_WD;
131:    SUBSID.SUR_WD = SAVE_RLG.SUR_WD;
132:    SUBSID.YR_IMP = SAVE_RLG.YR_IMP;

133: SURFACE: K = SAVE_RLG.SURTYP;
134:    CALL SRTYPRA(K);
135:    SUBSID.SURTYP = SRTYP(K);

136: REWRITE: REWRITE FILE(SUFFSUB) FROM(STRING_SUBSID);
137:    PRINTER = STRING_SUBSID;
138:    CALL PRINTX(F(1));


139: /* READ THE NEXT SUBSIDIARY RECORD TO BE UPDATED */
140:    READ FILE(SUFFSUB) SET(SUB_PTR);
141:    SUBSID_KEY = SUBSTR(STRING_SUBSID,2,13);
142:    IF SUBSID_KEY > ENDKEY THEN GO TO CLOSE;
143:    ELSE GO TO CHECK;


144: /* CLOSE FILES */
```

```
SYSR: PROCEDURE(PARM) OPTIONS(MAIN);


145: CLOSE: CLOSE FILE(ROADLOG);
146:    CLOSE FILE(SUFFSUB);
147:    CALL EXIT(PARM);
148:    END SYSR;
```

PHASE=TRAFFIC:

Member Name	SYST	
Language	PL/I	
Subroutine	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- "dump" listing
	TRAFFIC	-- Traffic file
	SUFFSUB	-- Sufficiency Report file
	TRUMILE	-- True Mileage file
	SUFWORK	-- Scratch file
Instruction	1 - 4	"SYST"

This program calculates a weighted average daily traffic, a design hour volume, a percent commercial, and the current average daily traffic for each sufficiency section in the current average daily traffic for each sufficiency section in the Sufficiency Report file. The data needed to make these calculations is stored in the Traffic file, the Traffic Summary file, and the True Mileage File.

The SYST program listing follows:

```
SYTT:PROCEDURE(PARM) OPTIONS(MAIN);

  1: SYTT:PROCEDURE(PARM) OPTIONS(MAIN);
  2:    DECLARE
  3:    SUFWORK FILE RECORD,
  4:    SUFFSUB FILE RECORD KEYED ENV(INDEXED),
  5:    TRAFFIC FILE RECORD KEYED ENV(INDEXED GENKEY),
  6:    TRUMILE FILE RECORD KEYED ENV(INDEXED GENKEY),
  7:    BLANK STATIC                          CHAR(120) INIT(' '),
  8:    CURRENT_ADT(3)                        PIC'ZZZZZ',
  9:    CUREENT_DHV                           PIC'VZZZZ',
 10:    CURRENT_MILEAGE                       DEC FIXED (7,3),
 11:    CURRENT2_ADT(3)                       PIC'ZZZZZ',
 12:    CURENT2_DHV                           PIC'VZZZZ',
 13:    ENDKEY                                CHAR(13) DEF INSTR POS(56),
 14:    F(0:9) STATIC                         PIC'Z'
 15:                                          INIT(0,1,2,3,4,5,6,7,8,9),
 16:    INSTR                                 CHAR(80) EXT,
 17:    PARM                                  CHAR(100),
 18:    PRINTER                               CHAR(132) EXT,
 19:    1 SAVE_SUBSID BASED(SAVE_SUBSID_PTR) LIKE SUBSID,
 20:    SAVE_SUBSID_PTR                       PTR,
 21:    1 SAVE_TRF LIKE TRF BASED(SAVE_TRF_PTR),
 22:    SAVE_TRF_PTR                          PTR,
 23:    STARTKEY                              CHAR(13) DEF INSTR POS(40),
 24:    STRING_SAVE_SUBSID BASED(SAVE_SUBSID_PTR) CHAR(120),
 25:    STRING_SAVE_TRF BASED(SAVE_TRF_PTR) CHAR(80),
 26:    STRING_SUBSID BASED(SUBSID_PTR) CHAR(120),
 27:    STRING_TRF BASED(TRF_PTR) CHAR(80),
 28:    STRING_TRFA BASED(TRFA_PTR) CHAR(80),
 29:    1 SUBSID BASED(SUBSID_PTR),
 30:         5 DUMMY1                         CHAR(1),
 31:         5 KEY,
 32:              10 SYSTEM                   PIC'Z',
 33:              10 ROUTE_#                  PIC'ZZZ',
 34:              10 MILEPOST                 PIC'ZZZ',
 35:              10 FRACTION                 PIC'+ZV.ZZZ',
 36:         5 REMARK                         CHAR(1),
 37:         5 DESCR                          CHAR(18),
 38:         5 COUNTY_#                       PIC'ZZ',
 39:         5 FINANCIAL_DISTRICT             PIC'ZZ',
 40:         5 YR_BLT                         PIC'ZZ',
 41:         5 YR_IMP                         PIC'ZZ',
 42:         5 SUR_WD                         PIC'ZZ',
 43:         5 RDY_WD                         PIC'ZZ',
 44:         5 SURTYP                         PIC'ZZZ',
 45:         5 SECTION_LENGTH                 PIC'ZZZVZZZ',
 46:         5 ADT                            PIC'ZZZZZ',
 47:         5 DHV                            PIC'ZZZZ',
 48:         5 PERCENT_TRUCKS                 PIC'ZZ',
 49:         5 SERVICE_VOL                    PIC'ZZZZ',
 50:         5 #_ACCIDENTS                    PIC'ZZ',
 51:         5 FOUNDATION                     PIC'ZZ',
 52:         5 SURFACE                        PIC'ZZ',
 53:         5 DRAINAGE                       PIC'ZZ',
 54:         5 SAFTEY_RATING                  PIC'ZZ',
 55:         5 CAPACITY_RATING                PIC'ZZ',
 56:         5 TOTAL_RATING                   PIC'ZZZ',
 57:         5 ADJ_RATING                     PIC'ZZZ',
 58:         5 DEFICIENT_MILEAGE              PIC'ZZVZZ',
```

```
SYTT:PROCEDURE(PARM) OPTIONS(MAIN);

 59:          5 DESIGN_SPEED                  PIC'ZZ',
 60:          5 TERRAIN                       PIC'Z',
 61:          5 AVG_SPEED                     PIC'ZZ',
 62:          5 SIGHT_DIST                    PIC'ZZ',
 63:          5 STOP_DIST                     PIC'VZZ',
 64:          5 CURVES                        PIC'ZZ',
 65:          5 BRIDGES                       PIC'Z',
 66:          5 #_LANES                       PIC'Z',
 67:          5 DIVIDED_CODE                  CHAR(1),
 68:          5 CITY_#                        PIC'ZZZ',
 69:          5 CURRENT_SECTION_ADT           PIC'ZZZZZ',
 70:          5 DUMMY5                        CHAR(7),
 71:       SUBSID_PTR                         PTR,
 72:       1 TRF BASED(TRF_PTR),
 73:          2 DUMMY                         CHAR(1),
 74:          2 KEY                           CHAR(13),
 75:          2 ROUTE_#                       DEC FIXED (3,0),
 76:          2 MILEPOST                      DEC FIXED (3,0),
 77:          2 FRACTION                      DEC FIXED (5,3),
 78:          2 ACTUAL_ESTIMATED              CHAR(1),
 79:          2 REMARK                        CHAR(1),
 80:          2 DATA(4),
 81:             3 YEAR                       DEC FIXED (3,0),
 82:             3 ADT                        DEC FIXED (5,0),
 83:             3 OUT_OF_STATE               DEC FIXED (3,3),
 84:             3 PICKUPS                    DEC FIXED (3,3),
 85:             3 COMMERCIAL                 DEC FIXED (3,3),
 86:          2 FUTURE_FACTOR                 DEC FIXED (3,3),
 87:          2 DHV                           DEC FIXED (3,3),
 88:          2 DATA_OF_UPDATE                CHAR(6),
 89:          2 DUMMY2                        CHAR(2),
 90:       TRFCHAR                            CHAR(80),
 91:       TRF_FLAG(3)                        CHAR(1),
 92:       TRF_PTR                            PTR,
 93:       1 TRFA LIKE TRF BASED(TRFA_PTR),
 94:       TRFA_PTR                           PTR,
 95:       1 TRM BASED(TRM_PTR),
 96:          2 DELETE_CHAR                   CHAR(1),
 97:          2 KEY,
 98:             3 SYSTEM                     CHAR(1),
 99:             3 ROUTE_#                    PIC'999',
100:             3 MILEPOST                   PIC'999',
101:          2 TRUE_MILEAGE                  DEC FIXED (7,3),
102:          2 DATE_OF_UPDATE                DEC FIXED (7,0),
103:       TRM_PTR                            PTR,
104:       TRUE_ARRAY(0:1000) STATIC          DEC FIXED (7,3),
105:       TRUE_ROUTE                         CHAR(4),
106:       VEH_MILES(3)                       DEC FIXED (14,3);
107:       ON ENDFILE(SUFFSUB) GO TO COPY;
108:       ON ENDFILE(SUFWORK) GO TO CLOSE;
109:       CALL INIT(PARM);
110:       OPEN FILE(SUFFSUB) INPUT SEQL;
111:       OPEN FILE(TRAFFIC) INPUT SEQL;
112:       OPEN FILE(TRUMILE) INPUT SEQL;
113:       OPEN FILE(SUFWORK) OUTPUT SEQL;
114:       TRUE_ARRAY(0) = 0;
115:       READ FILE(SUFFSUB) SET(SUBSID_PTR) KEY(STARTKEY);
116:       READ FILE(TRAFFIC) SET(TRFA_PTR) KEY(SUBSID.SYSTEM||SUBSID.ROUTE_#);
```

```
SYTT:PROCEDURE(PARM) OPTIONS(MAIN);

117:     SAVE_SUBSID_PTR = ADDR(BLANK);
118:     TRUE_ROUTE = '     ';
119: LOOP: STRING_SAVE_SUBSID = STRING_SUBSID;
120:     READ FILE(SUFFSUB) SET(SUBSID_PTR);
121:     IF ENDKEY < SUBSTR(STRING_SAVE_SUBSID,2,13) THEN GO TO COPY;
122:     IF SAVE_SUBSID.REMARK = 'U' |
123:        SAVE_SUBSID.REMARK = 'M' |
124:        SAVE_SUBSID.REMARK = 'N' |
125:        SAVE_SUBSID.REMARK = 'C' |
126:        SAVE_SUBSID.DESCR = 'END OF ROUTE      ' THEN DO;
127:           WRITE FILE(SUFWORK) FROM(STRING_SAVE_SUBSID);
128:           PRINTER = STRING_SAVE_SUBSID;
129:           CALL PRINTX(F(1));
130:           GO TO LOOP;
131:           END;
132:     IF TRUE_ROUTE ¬= (SAVE_SUBSID.SYSTEM||SAVE_SUBSID.ROUTE_#) THEN DO;
133:        TRUE_ROUTE = SAVE_SUBSID.SYSTEM||SAVE_SUBSID.ROUTE_#;
134:        READ FILE(TRUMILE) SET(TRM_PTR) KEY(TRUE_ROUTE);
135:        DO WHILE (TRM.ROUTE_# = SAVE_SUBSID.ROUTE_#);
136:           TRUE_ARRAY(TRM.MILEPOST) = TRUE_MILEAGE;
137:           READ FILE(TRUMILE) SET(TRM_PTR);
138:           END;
139:     END;


140:     DO WHILE (TRFA.KEY <= SUBSTR(STRING_SAVE_SUBSID,2,13));
141:     TRFCHAR = STRING_TRFA;
142:     TRF_PTR = ADDR(TRFCHAR);
143:     READ FILE(TRAFFIC) SET(TRFA_PTR);
144:     END;

145:     SAVE_TRF_PTR = TRF_PTR;
146:     DO I = 1 TO 3;
147:        CURRENT_ADT(I) = SAVE_TRF.DATA(I).ADT;
148:        END;
149:     CURRENT_DHV = SAVE_TRF.DHV;
150:     IF (TRF.KEY ¬= SUBSTR(STRING_SAVE_SUBSID,2,13)) THEN DO;
151:        X1 = (TRUE_ARRAY(SAVE_SUBSID.MILEPOST) + SAVE_SUBSID.FRACTION)
152:        - (TRUE_ARRAY(SAVE_TRF.MILEPOST) + SAVE_TRF.FRACTION);
153:        X2 = (TRUE_ARRAY(TRFA.MILEPOST) + TRFA.FRACTION) -
154:        (TRUE_ARRAY(SAVE_SUBSID.MILEPOST) + SAVE_SUBSID.FRACTION);
155:        DO I = 1 TO 3;
156:           CURRENT_ADT(I) = SAVE_TRF.DATA(I).ADT + ((TRFA.DATA(I).ADT -
157:           SAVE_TRF.DATA(I).ADT) * (X1/(X1+X2)));
158:           END;
159:        CURRENT_DHV = SAVE_TRF.DHV + (TRFA.DHV - TRF.DHV)*(X1/(X1+X2));
160:        END;
161:     VEH_MILES = 0;
162:     CURRENT_MILEAGE = (TRUE_ARRAY(SAVE_SUBSID.MILEPOST) +
163:     SAVE_SUBSID.FRACTION);
164:     SAVE_SUBSID.DHV = CURRENT_ADT(3) * CURRENT_DHV;
165:     SAVE_SUBSID.PERCENT_TRUCKS = SAVE_TRF.DATA(3).COMMERCIAL*100;
166:     IF SAVE_TRF.DATA(1).ADT = 0 THEN TRF_FLAG(1) = 'B';
167:     IF SAVE_TRF.DATA(2).ADT = 0 THEN TRF_FLAG(2) = 'B';


168:     DO WHILE ( TRFA.KEY <= SUBSTR(STRING_SUBSID,2,13));
169:        TRFCHAR = STRING_TRFA;
```

```
SYTT:PROCEDURE(PARM) OPTIONS(MAIN);

170:        TRF_PTR = ADDR(TRFCHAR);
171:     READ FILE(TRAFFIC) SET(TRFA_PTR);
172:        X1 = (TRUE_ARRAY(TRF.MILEPOST) + TRF.FRACTION) -
173:        CURRENT_MILEAGE;
174:        DO I = 1 TO 3;
175:           VEH_MILES(I) = VEH_MILES(I) + (CURRENT_ADT(I) +
176:           TRF.DATA(I).ADT) * (X1/2);
177:           CURRENT_ADT(I) = TRF.DATA(I).ADT;
178:           END;
179:        CURRENT_DHV = TRF.DHV;
180:        CURRENT_MILEAGE = (TRUE_ARRAY(TRF.MILEPOST) +
181:        TRF.FRACTION);
182:        IF TRF.DATA(1).ADT = 0 THEN TRF_FLAG(1) = 'B';
183:        IF TRF.DATA(2).ADT = 0 THEN TRF_FLAG(2) = 'B';
184:        IF (TRF.DATA(3).ADT * TRF.DHV ) > SAVE_SUBSID.DHV THEN DO;
185:           SAVE_SUBSID.DHV = TRF.DATA(3).ADT * TRF.DHV;
186:           SAVE_SUBSID.PERCENT_TRUCKS = TRF.DATA(3).COMMERCIAL*100;
187:           END;
188:        END;


189:      IF TRF.KEY ¬= SUBSTR(STRING_SUBSID,2,13) THEN DO;
190:        X1 = (TRUE_ARRAY(SUBSID.MILEPOST) + SUBSID.FRACTION) -
191:           (TRUE_ARRAY(TRF.MILEPOST) + TRF.FRACTION);
192:        X2 = (TRUE_ARRAY(TRFA.MILEPOST) + TRFA.FRACTION) -
193:           (TRUE_ARRAY(SUBSID.MILEPOST) + SUBSID.FRACTION);
194:        DO I = 1 TO 3;
195:           CURRENT2_ADT(I) = TRF.DATA(I).ADT + (TRFA.DATA(I).ADT
196:           -  TRF.DATA(I).ADT) * (X1/(X1+X2));
197:           END;
198:        CURRENT2_DHV = TRF.DHV + (TRFA.DHV -TRF.DHV) * (X1/(X1+X2));
199:        X1 = (TRUE_ARRAY(SUBSID.MILEPOST) + SUBSID.FRACTION) -
200:           CURRENT_MILEAGE;
201:        DO I = 1 TO 3;
202:           VEH_MILES(I) = VEH_MILES(I) + ((CURRENT_ADT(I) +
203:           CURRENT2_ADT(I)) * X1/2);
204:           END;


205:        IF (CURRENT2_DHV * CURRENT2_ADT(3))> SAVE_SUBSID.DHV THEN DO;
206:           SAVE_SUBSID.DHV = CURRENT2_DHV * CURRENT2_ADT(3);
207:           SAVE_SUBSID.PERCENT_TRUCKS = TRF.DATA(3).COMMERCIAL * 100;
208:           END;
209:        END;


210:     IF TRF_FLAG(2) = 'B' THEN SAVE_SUBSID.ADT = VEH_MILES(3) /
211:        SAVE_SUBSID.SECTION_LENGTH;
212:        ELSE IF TRF_FLAG(1) = 'B' THEN SAVE_SUBSID.ADT = (VEH_MILES(3) +
213:           VEH_MILES(2))/(SAVE_SUBSID.SECTION_LENGTH * 2);
214:        ELSE SAVE_SUBSID.ADT = (VEH_MILES(3) + VEH_MILES(2) +
215:           VEH_MILES(1)) / (SAVE_SUBSID.SECTION_LENGTH * 3);
216:     SAVE_SUBSID.CURRENT_SECTION_ADT = VEH_MILES(3)/
217:      SAVE_SUBSID.SECTION_LENGTH;
218:     TRF_FLAG = ' ';


219: PRINT: WRITE FILE(SUFWORK) FROM(STRING_SAVE_SUBSID);
```

```
SYTT:PROCEDURE(PARM) OPTIONS(MAIN);

220:     PRINTER = STRING_SAVE_SUBSID;
221:     CALL PRINTX(F(1));
222:     GO TO LOOP;


223: COPY: CLOSE FILE(SUFWORK);
224:     CLOSE FILE(SUFFSUB);
225:     OPEN FILE(SUFWORK) INPUT SEQL;
226:     OPEN FILE(SUFFSUB) OUTPUT SEQL;


227: COPY2: READ FILE(SUFWORK) SET(SUBSID_PTR);
228:     WRITE FILE(SUFFSUB) FROM(STRING_SUBSID)
229:          KEYFROM(SUBSTR(STRING_SUBSID,2,13));
230:     GO TO COPY2;


231: CLOSE: CLOSE FILE(SUFFSUB);
232:     CLOSE FILE(TRAFFIC);
233:     CLOSE FILE(TRUMILE);
234:     CLOSE FILE(SUFWORK);
235:     CALL EXIT(PARM);
236:     END SYTT;
```

PHASE=ACCIDENT:

```
Member Name . . . . . . . SYSA
Language  . . . . . . . . PL/I
Subroutine  . . . . . . . PRINTX1
Files . . . . . . . . . . SYSPRINT -- IBM messages
                          PRINTER  -- "dump" listing
                          SUFFSUB  -- Sufficiency Report file
                          ACCDIRI  -- Accident Directoy file
Instruction . . . . . . .  1 - 4   "SYSA"
```

This phase of CREATE-SUFFSUB is designed to sum up the number of accidents which have occurred within each sufficiency section over the past three years. The number of accidents is retrieved from the Accident Directory file and placed in the Sufficiency Report file.

(******* PLEASE NOTE -- The program listed on the following pages does not use the Accident Directory file. At the time the Sufficiency Subsystem was built the Accident Directory file did not exist. Therefore, this temporary program accesses an Accident file containing the average number of accidents for each sufficiency section. This program will be updated to retrieve data from the Accident Directory file as soon as sufficient accident data has been stored in the file. *******)

```
SYSA: PROCEDURE(PARM) OPTIONS(MAIN);

 1: SYSA: PROCEDURE(PARM) OPTIONS(MAIN);


 2:    DECLARE ACIDENT FILE RECORD KEYED ENV(INDEXED),
 3:    SUFFSUB FILE RECORD KEYED ENV(INDEXED);
 4:    DECLARE
 5:    1 ACC BASED(ACC_PTR),
 6:         5 DUMMY1                          CHAR(1),
 7:         5 KEY                             CHAR(9),
 8:         5 DUMMY2                          CHAR(2),
 9:         5 #_ACCIDENTS                     PIC'ZZ',
10:    ACC_PTR                                PTR,
11:    ACC_KEY                                CHAR(9),
12:    ENDKEY                                 CHAR(13) DEF INSTR POS(56),
13:    F(0:9)                                 PIC'Z'
14:                                           INIT(0,1,2,3,4,5,6,7,8,9),
15:    INSTR                                  CHAR(80) EXT,
16:    PARM                                   CHAR(100),
17:    PRINTER                                CHAR(132) EXT,
18:    STARTKEY                               CHAR(13) DEF INSTR POS(40),
19:    STRING_SUBSID BASED(SUB_PTR) CHAR(120),
20:    1 SUBSID BASED(SUB_PTR),
21:         5 DUMMY1                          CHAR(1),
22:         5 SYSTEM                          CHAR(1),
23:         5 ROUTE_#                         PIC'ZZZ',
24:         5 MILEPOST                        PIC'ZZZ',
25:         5 PLUS                            PIC'+Z.',
26:         5 FRACTION                        PIC'VZZZ',
27:         5 REMARK                          CHAR(1),
28:         5 DESCR                           CHAR(18),
29:         5 DUMMY2                          CHAR(36),
30:         5 #_ACCIDENTS                     PIC'ZZ',
31:    SUB_PTR                                PTR;


32:    CALL INIT(PARM);


33:    ON ENDFILE(SUFFSUB) GO TO CLOSE;


34:    ON ENDFILE(ACIDENT) GO TO CLOSE;


35: /* OPEN FILES */

36:     OPEN FILE(SUFFSUB) UPDATE SEQL;
37:    OPEN FILE(ACIDENT) INPUT SEQL;


38: LOOP: READ FILE(SUFFSUB) SET(SUB_PTR);
39:    ACC_KEY = SUBSID.SYSTEM ||
40:         SUBSID.ROUTE_# ||
41:         SUBSID.MILEPOST ||
42:         SUBSID.FRACTION;
43:    IF ENDKEY <= (SUBSID.SYSTEM||SUBSID.ROUTE_#||SUBSID.MILEPOST||
44:    SUBSID.PLUS||SUBSID.FRACTION) THEN GO TO CLOSE;
45:    IF SUBSID.REMARK = 'M' |
```

```
SYSA: PROCEDURE(PARM) OPTIONS(MAIN);

 46:          SUBSID.REMARK = 'U' |
 47:          SUBSID.REMARK = 'N' |
 48:          SUBSID.REMARK = 'C' |
 49:          SUBSID.DESCR = 'END OF ROUTE       ' THEN GO TO PRINT;
 50:     READ FILE(ACIDENT) SET(ACC_PTR) KEY(ACC_KEY);
 51:     SUBSID.#_ACCIDENTS = ACC.#_ACCIDENTS;
 52:     REWRITE FILE(SUFFSUB) FROM(STRING_SUBSID);
 53: PRINT: PRINTER = STRING_SUBSID;
 54:     CALL PRINTX(F(1));
 55:     GO TO LOOP;


 56: CLOSE: CLOSE FILE(SUFFSUB);
 57:     CLOSE FILE(ACIDENT);
 58:     CALL EXIT(PARM);
 59:     END SYSA;
```

PHASE=CALCULATIONS:

Member Name	SYSC	
Language	PL/I	
Subroutine	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- "dump" listing
	SUFFSUB	-- Sufficiency Report file
	SUFWORK	-- Scratch file
	SUFFTBL	-- Table of rating factors
Instruction	1 - 4	"SYSC"

SYSC is the final phase of CREATE-SUFFSUB. This program uses the information stored in the Sufficiency Report file by the Sufficiency, Roadlog, Traffic, and Accident Phases. From this information it calculates the Total Rating, Safety Rating, Adjusted Rating, Capacity Rating, Deficient Mileage, and Service Volume for each sufficiency section.

The SYSC program listing follows:

```
SYSC: PROCEDURE(PARM) OPTIONS(MAIN);

 1: SYSC: PROCEDURE(PARM) OPTIONS(MAIN);
 2:    DECLARE
 3:    SUFFSUB FILE RECORD KEYED ENV(INDEXED),
 4:    TABLE FILE RECORD,
 5:    TRAFSUM FILE RECORD KEYED ENV(GENKEY INDEXED),
 6:    ADJ_FACTOR(4,5,15)                    PIC'ZZVZZ',
 7:    CONSTANT                              FIXED(3,0),
 8:    ENDKEY                                CHAR(13) DEF INSTR POS(56),
 9:    F(0:9)                                PIC'Z'
10:                                          INIT(0,1,2,3,4,5,6,7,8,9),
11:    FACTOR                                PIC'9V99',
12:    FOUR_LANE_COMM                        PIC'9V99',
13:    FOUR_LANE_SPEED(3,4)                  PIC'ZZVZZ',
14:    FOUR_LANE_WID(5)                      PIC'ZZVZZ',
15:    HOURLY_VOL(16,11)                     PIC'ZZZZ',
16:    1 IN BASED(IN_PTR),
17:         5 HOURLY_VOLUME(11)              PIC'ZZZZ',
18:    IN_PTR                                PTR,
19:    1 IN1 BASED(IN1_PTR),
20:         5 ADJ(5)                         PIC'ZZVZZ',
21:    IN1_PTR                               PTR,
22:    1 IN2 BASED(IN2_PTR),
23:         5 ADJ(3)                         PIC'ZZVZZ',
24:    IN2_PTR                               PTR,
25:    1 IN3 BASED(IN3_PTR),
26:         5 ADJ(4)                         PIC'ZZVZZ',
27:    IN3_PTR                               PTR,
28:    1 IN4 BASED(IN4_PTR),
29:         5 ADJ(5)                         PIC'ZZVZZ',
30:    IN4_PTR                               PTR,
31:    INSTR                                 CHAR(80) EXT,
32:    LANE_WIDTH                            PIC'99',
33:    PARM                                  CHAR(100),
34:    PRINTER                               CHAR(132) EXT,
35:    SHOULDER                              PIC'99',
36:    STARTKEY                              CHAR(13) DEF INSTR POS(40),
37:    STRING_SUBSID BASED(SUB_PTR) CHAR(120),
38:    SUBSCRIPT(4,7)                        PIC'ZZ',
39:    1 SUBSID_KEY BASED(SUB_PTR),
40:         5 DUMMY1                         CHAR(1),
41:         5 KEY                            CHAR(13),
42:    1 SUBSID BASED(SUB_PTR),
43:         5 DUMMY1                         CHAR(1),
44:         5 KEY,
45:              10 SYSTEM                   CHAR(1),
46:              10 ROUTE_#                  PIC'ZZZ',
47:              10 MILEPOST                 PIC'ZZZ',
48:              10 FRACTION                 PIC'+ZV.ZZZ',
49:         5 REMARK                         CHAR(1),
50:         5 DESCR                          CHAR(18),
51:         5 COUNTY_#                       PIC'ZZ',
52:         5 FINANCIAL_DISTRICT             PIC'ZZ',
53:         5 YR_BLT                         PIC'ZZ',
54:         5 YR_IMP                         PIC'ZZ',
55:         5 SUR_WD                         PIC'ZZ',
56:         5 RDY_WD                         PIC'ZZ',
57:         5 SURTYP                         PIC'ZZZ',
58:         5 SECTION_LENGTH                 PIC'ZZZVZZZ',
```

```
SYSC: PROCEDURE(PARM) OPTIONS(MAIN);

 59:          5 ADT                          PIC'ZZZZZ',
 60:          5 DHV                          PIC'ZZZZ',
 61:          5 PERCENT_TRUCKS               PIC'ZZ',
 62:          5 SERVICE_VOL                  PIC'ZZZZ',
 63:          5 #_ACCIDENTS                  PIC'ZZ',
 64:          5 FOUNDATION                   PIC'ZZ',
 65:          5 SURFACE                      PIC'ZZ',
 66:          5 DRAINAGE                     PIC'ZZ',
 67:          5 SAFTEY_RATING                PIC'ZZ',
 68:          5 CAPACITY_RATING              PIC'ZZ',
 69:          5 TOTAL_RATING                 PIC'ZZZ',
 70:          5 ADJ_RATING                   PIC'ZZZ',
 71:          5 DEFICIENT_MILEAGE            PIC'ZZVZZ',
 72:          5 DESIGN_SPEED                 PIC'ZZ',
 73:          5 TERRAIN                      PIC'Z',
 74:          5 AVG_SPEED                    PIC'ZZ',
 75:          5 SIGHT_DIST                   PIC'VZZ',
 76:          5 STOP_DIST.                   PIC'ZZ',
 77:          5 CURVES                       PIC'ZZ',
 78:          5 BRIDGES                      PIC'Z',
 79:          5 #_LANES                      PIC'Z',
 80:          5 DIVIDED_CODE                 CHAR(1),
 81:          5 CITY_#                       PIC'ZZZ',
 82:          5 CURRENT_SECTION_ADT          PIC'ZZZZZ',
 83:     SUB_PTR                             PTR,
 84:     1 SUM BASED(SUM_PTR),
 85:          5 DUMMY1                       CHAR(15),
 86:          5 RURAL_MILEAGE                DEC FIXED(7,3),
 87:          5 DUMMY12                      CHAR(48),
 88:          5 ALL_VEH                      DEC FIXED(11,3),
 89:     SUM_PTR                             PTR,
 90:     SYSTEM_ADT                          PIC'99999V9',
 91:     SYSTEM_ADTS(3)                      PIC'ZZZZZVZ',
 92:     SYSTEM_KEY(3)                       CHAR(4)
 93:                                         INIT('I999','P999','S999'),
 94:     TEMP                                PIC'ZZZZZZVZ',
 95:     THREE_LANE(4,5,3)                   PIC'ZVZZ',
 96:     TRUCK_TERRAIN                       PIC'99',
 97:     VEH_ACC_RATE                        PIC'ZZZV9';


 98:     CALL INIT(PARM);


 99:     ON ENDFILE(SUFFSUB) GO TO CLOSE;


100: /* READ SUFFICIENCY TABLES */
101:     OPEN FILE(TABLE) INPUT SEQL TITLE('SUFFTBL');
102:     DO I = 1 TO 16;
103:          READ FILE(TABLE) SET(IN_PTR);
104:          HOURLY_VOL(I,*) = IN.HOURLY_VOLUME;
105:          END;
106:     DO I = 1 TO 4;
107:          DO K = 1 TO 15;
108:          READ FILE(TABLE) SET(IN1_PTR);
109:          ADJ_FACTOR(I,*,K) = IN1.ADJ;
110:          END;
```

```
SYSC: PROCEDURE(PARM) OPTIONS(MAIN);

111:          END;
112:     DO K = 1 TO 4;
113:     DO I = 1 TO 5;
114:          READ FILE(TABLE) SET(IN2_PTR);
115:          THREE_LANE(K,I,*) = IN2.ADJ;
116:          END;
117:          END;
118:     DO I = 1 TO 3;
119:          READ FILE(TABLE) SET(IN3_PTR);
120:          FOUR_LANE_SPEED(I,*) = IN3.ADJ;
121:          END;
122:     READ FILE(TABLE) SET(IN4_PTR);
123:     FOUR_LANE_WID = IN4.ADJ;
124:     CLOSE FILE(TABLE);
125:     L = 0;
126:     DO K = 1 TO 4;
127:     DO I = 1 TO 7 WHILE(K=1),
128:          3 TO 7 WHILE(K=2),
129:          5 TO 7 WHILE(K=3),
130:          7 WHILE(K=4);
131:          L = L + 1;
132:          SUBSCRIPT(K,I)=L;
133:          END;
134:          END;
135:     OPEN FILE(TRAFSUM) INPUT SEQL;
136:     DO I = 1 TO 3;
137:          READ FILE(TRAFSUM) SET(SUM_PTR) KEY(SYSTEM_KEY(I));
138:          SYSTEM_ADTS(I) = SUM.ALL_VEH/SUM.RURAL_MILEAGE;
139:          END;
140:     CLOSE FILE(TRAFSUM);
141:     CONSTANT = 100;
142:     OPEN FILE(SUFFSUB) UPDATE SEQL;
143:     READ FILE(SUFFSUB) SET(SUB_PTR) KEY(STARTKEY);
144:     IF SUBSID.KEY.SYSTEM = 'I' THEN SYSTEM_ADT=SYSTEM_ADTS(1);
145:          ELSE IF SUBSID.KEY.SYSTEM='P' THEN SYSTEM_ADT=SYSTEM_ADTS(2);
146:          ELSE SYSTEM_ADT= SYSTEM_ADTS(3);
147:     GO TO CALC;


148: START: READ FILE(SUFFSUB) SET(SUB_PTR);
149:     IF (ENDKEY <= SUBSID_KEY.KEY) THEN GO TO CLOSE;
150: CALC: IF SUBSID.REMARK = 'C' |
151:          SUBSID.REMARK = 'M' |
152:          SUBSID.REMARK = 'U' |
153:          SUBSID.REMARK = 'N' |
154:          SUBSID.DESCR = 'END OF ROUTE      '
155:          THEN GO TO START;
156:     IF SUBSID.#_LANES < 2 THEN SUBSID.#_LANES = 2;
157:     LANE_WIDTH = SUBSID.SUR_WD / SUBSID.#_LANES;
158:     IF LANE_WIDTH < 9 THEN LANE_WIDTH = 9;
159:     IF LANE_WIDTH > 12 THEN LANE_WIDTH = 12;
160:     SHOULDER = ABS((SUBSID.RDY_WD - SUBSID.SUR_WD) / 2 - 5);
161:     IF SHOULDER > 5 THEN SHOULDER = 5;
162:     IF SUBSID.#_LANES > 4 THEN SUBSID.#_LANES = 4;


163:     IF SUBSID.#_LANES = 4 THEN DO;
```

```
SYSC: PROCEDURE(PARM) OPTIONS(MAIN);


164:     /* THE FOURLANE COMMERCIAL FACTOR IS COMPUTED BY THE FORMULA
165:     GIVEN ON PAGE 261 OF THE 1965 HIGHWAY CAPACITY MANUAL */
166:          IF SUBSID.TERRAIN = 3 THEN K=8;
167:          ELSE K = SUBSID.TERRAIN * 2;
168:       FOUR_LANE_COMM = 100/ (100 - SUBSID.PERCENT_TRUCKS +
169:          SUBSID.PERCENT_TRUCKS * K);
170:          IF SUBSID.DIVIDED_CODE = 'D' THEN
171:          SHOULDER = (SUBSID.RDY_WD - SUBSID.SUR_WD) / 4 + 1;
172:          ELSE SHOULDER = (SUBSID.RDY_WD - SUBSID.SUR_WD)/2 + 1;
173:          IF SHOULDER > 5 THEN SHOULDER = 5;
174:          FACTOR = FOUR_LANE_SPEED(SUBSID.AVG_SPEED/10 - 3.4,
175:          SUBSID.DESIGN_SPEED/10 - 2.4) * FOUR_LANE_WID(SHOULDER);
176:          SUBSID.SERVICE_VOL = 8000 * FOUR_LANE_COMM * FACTOR;
177:          GO TO CALC3;
178:          END;

179:     IF SUBSID.#_LANES = 3 THEN DO;
180:          FACTOR = THREE_LANE(LANE_WIDTH - 8,SHOULDER,SUBSID.TERRAIN);
181:          GO TO CALC2;
182:          END;

183:     IF SUBSID.PERCENT_TRUCKS = 0 THEN
184:     TRUCK_TERRAIN = 1 + (SUBSID.TERRAIN - 1) * 5;
185:     ELSE TRUCK_TERRAIN = SUBSID.PERCENT_TRUCKS / 5 + .99 +
186:     (SUBSID.TERRAIN - 1) *5;
187:     FACTOR = ADJ_FACTOR(LANE_WIDTH - 8,SHOULDER,TRUCK_TERRAIN);

188: CALC2:SUBSID.SERVICE_VOL = FACTOR * HOURLY_VOL(SUBSCRIPT
189:     (SUBSID.DESIGN_SPEED/10-3,SUBSID.AVG_SPEED/5 - 7),
190:     SUBSID.SIGHT_DIST * 10 + 1.99);

191: CALC3: SUBSID.CAPACITY_RATING = 30 - (15 * SUBSID.DHV /
192:     SUBSID.SERVICE_VOL) + .5;
193:     VEH_ACC_RATE = (SUBSID.#_ACCIDENTS * 10000000) /
194:     ( 365 * SUBSID.ADT * CONSTANT);
195:     TEMP = SUBSID.STOP_DIST + VEH_ACC_RATE + SUBSID.CURVES +
196:     SUBSID.BRIDGES;
197:     IF TEMP = 0 THEN SUBSID.SAFTEY_RATING = 20;
198:          ELSE DO;
199:          SUBSID.SAFTEY_RATING = 2 * SUBSID.SECTION_LENGTH /
200:          (SUBSID.STOP_DIST + VEH_ACC_RATE + SUBSID.CURVES +
201:          SUBSID.BRIDGES) + .5;
202:          IF SUBSID.SAFTEY_RATING > 20 THEN SUBSID.SAFTEY_RATING = 20;
203:          END;
204:     SUBSID.TOTAL_RATING = SUBSID.SAFTEY_RATING + SUBSID.CAPACITY_RATING
205:     + SUBSID.FOUNDATION + SUBSID.DRAINAGE + SUBSID.SURFACE;
206:     SUBSID.ADJ_RATING = SUBSID.TOTAL_RATING + ((SUBSID.TOTAL_RATING**2
207:     - 100 * SUBSID.TOTAL_RATING) / (50 * LOG10(SYSTEM_ADT)))
208:     * (LOG10(SUBSID.CURRENT_SECTION_ADT) - LOG10(SYSTEM_ADT));
209:     DEFICIENT_MILEAGE = (100 - SUBSID.TOTAL_RATING) *
210:     SUBSID.SECTION_LENGTH/100 + .05;


211: REWRITE: REWRITE FILE(SUFFSUB) FROM(STRING_SUBSID);
212:     PRINTER = STRING_SUBSID;
213:     CALL PRINTX(F(1));
214:     GO TO START;
```

```
SYSC: PROCEDURE(PARM) OPTIONS(MAIN);


215: /* CLOSE FILES */

216: CLOSE: CLOSE FILE(SUFFSUB);
217:    CALL EXIT(PARM);
218:    END SYSC;
```

LIST-BY-SECTION --

Member Name	SKS	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- SKS output
	SUFFSUB	-- Sufficiency Report file
	CNTYTBL	-- Table of county names
	CITYTBL	-- Table of city names
	ROADLOG	-- Roadlog file
Instruction	1 - 3	"SKS"
	40 - 43	Beginning route number
	56 - 59	Ending route number

List-by-Section provides a listing of the Sufficiency Report file in a report format. At the end of each Federal Aid Route a summary of the route length, the average adjusted rating for the route, and the total deficient mileage for the route are printed. If any sections of coincident roadway are encountered within the route, the program accesses the Roadlog file for a description of the coincident section.

The SKS program listing follows:

```
SK: PROCEDURE(PARM) OPTIONS(MAIN);

  1: SK: PROCEDURE(PARM) OPTIONS(MAIN);


  2: /***** DECLARATION OF VARIABLES *****/

  3: DECLARE
  4:    #_CITIES DEC FIXED (3) INIT(126),
  5:    #_COUNTIES DEC FIXED (2) INIT(56),
  6:    #_HDGS PIC'Z' DEF INSTR POS(72),
  7:    BLANKS CHAR(100) INIT(' '),
  8:    CARRIAGE DEC FIXED (1) INIT (1),
  9:    CITY(126) CHAR(18),
 10:    CITY_NAME CHAR(18) BASED (PTR_IN),
 11:    COUNTY_TABLE(0:56) CHAR(15),
 12:    1 COUNTY_STRUCTURE BASED(PTR_IN),
 13:       3 DUMMY1 CHAR(15),
 14:       3 COUNTY_NAME CHAR(15),
 15:    ENDKEY CHAR(13) DEF INSTR POS(56),
 16:    F(0:9) PIC'Z' INIT(0,1,2,3,4,5,6,7,8,9),
 17:    HEADING(9) CHAR(132) EXT,
 18:    1  IN BASED (PTR_IN),
 19:       3  DUM1 CHAR(1),
 20:       3  SYSTEM CHAR(1),
 21:       3  RT_# PIC'ZZZ',
 22:       3  MILEPOST PIC'999',
 23:       3  FRACTION PIC'+9V.999',
 24:       3  REMARK CHAR(1),
 25:       3  DESCRIP CHAR(18),
 26:       3  CNTY PIC'ZZ',
 27:       3  FINAN PIC'ZZ',
 28:       3  YR_BLT PIC'ZZ',
 29:       3  YR_IMP PIC'ZZ',
 30:       3  SURF_WIDTH PIC'ZZ',
 31:       3  RDY_WIDTH PIC'ZZ',
 32:       3  SURF_TYPE CHAR(3),
 33:       3  SECTION PIC'ZZZVZZZ',
 34:       3  ADT PIC'ZZZZZ',
 35:       3  DHV PIC'ZZZZ',
 36:       3  PERCENT_TRUCKS PIC'ZZ',
 37:       3  SERVICE_VOL PIC'ZZZZ',
 38:       3  #_ACCIDENTS PIC'ZZ',
 39:       3  FND_RATING PIC'ZZ',
 40:       3  SRF_RATING PIC'ZZ',
 41:       3  DRN_RATING PIC'ZZ',
 42:       3  SAF_RATING PIC'ZZ',
 43:       3  CAP_RATING PIC'ZZ',
 44:       3  TOT_RATING PIC'ZZZ',
 45:       3  ADJ_RATING PIC'ZZZ',
 46:       3  DEFIC PIC'ZZVZZ',
 47:       3 DESIGN_SPEED PIC'ZZ',
 48:       3  TERRAIN CHAR(1),
 49:       3  AVG_SPEED PIC'ZZ',
 50:       3  SIGHT_DIST PIC'ZZ',
 51:       3 STOP_DIST PIC'VZZ',
 52:       3  CURVES PIC'ZZ',
 53:       3  BRIDGES PIC'Z',
 54:       3  #_LANES PIC'Z',
 55:       3  DIVIDED_CODE CHAR(1),
```

```
SK: PROCEDURE(PARM) OPTIONS(MAIN);

 56:       3  CITY_# PIC'ZZZ',
 57:       3  CURRENT_SECTION_ADT PIC'ZZZZZ',
 58: 1 IN1 BASED (PTR_IN),
 59:       3  DUM CHAR(1),
 60:       3  KEY CHAR(13),
 61:    INSTR CHAR(80) EXT,
 62:    LINES CHAR(128) INIT((128)'-'),
 63:    1  OUT DEF STRING_OUT,
 64:       3  MIPOST CHAR(10),
 65:       3  DESCR CHAR(20),
 66:       3  COUNTY CHAR(16),
 67:       3  FINANCIAL_DIST PIC'ZZ',
 68:       3  YR_BLT PIC'ZZZZZ',
 69:       3  YR_IMP PIC'ZZZZ',
 70:       3  SECTN PIC'ZZZZZV.Z',
 71:       3  SURF_WIDTH PIC'ZZZZ',
 72:       3  RDY_WIDTH PIC'ZZZZB',
 73:       3  SURF_TYPE CHAR(3),
 74:       3  ADT PIC'ZZZZZZ',
 75:       3  DHV PIC'ZZZZZZ',
 76:       3  SERVICE_VOL PIC'ZZZZZZ',
 77:       3  FND_RATING PIC'ZZZZ9',
 78:       3  SRF_RATING PIC'ZZZ9',
 79:       3  DRN_RATING PIC'ZZZ9',
 80:       3  SAF_RATING PIC'ZZZ9',
 81:       3  CAP_RATING PIC'ZZZ9',
 82:       3  TOT_RATING PIC'ZZZ9',
 83:       3  ADJ_RATING PIC'ZZZ9',
 84:       3  DEFIC_MLGE PIC'ZZZZV.Z',
 85:    PAGE_POSITION PIC'ZZ' DEF INSTR POS(9),
 86:    PAGE_SIZE PIC'ZZ' DEF INSTR POS(7),
 87:    PRINTER CHAR(132) EXT,
 88:    PTR_IN PTR,
 89:    PTR_RLG PTR,
 90:    1  RLG BASED (PTR_RLG),
 91:       3 DUM CHAR(14),
 92:       3  DESCR CHAR(35),
 93:    ROADLOG FILE RECORD KEYED INPUT SEQL ENV(INDEXED),
 94:    ROUTE_DEFIC PIC'ZZZZVZ',
 95:    ROUTE_SECTN PIC'ZZZZVZZZ',
 96:    SAVE_RT_# PIC'ZZZ',
 97:    STARTKEY CHAR(13) DEF INSTR POS(40),
 98:    STRING_OUT CHAR(132),
 99:    SUBSID FILE RECORD KEYED INPUT SEQL ENV(INDEXED),
100:    SYS_DEFIC PIC'ZZZZVZ',
101:    SYS_SECTN PIC'ZZZZVZZZ',
102:    ZRT_# PIC'ZZZBB';


103: /***** PROGRAM INITIALIZATION *****/
104:       CALL INIT(PARM);

105:    /*** SET UP COLUMN HEADINGS *****/
106:    #_HDGS = 5;
107:       HEADING(3) = '   MILE        SECTION                    ' ||
108:       '    FIN  *YEAR*  SECTN  WIDTH  SUF              SERV  ' ||
109:       '*** SUFFICIENCY RATINGS ***  DEFIC';
110:       HEADING(4) = '   POST      DESCRIPTION                    ' ||
```

```
SK: PROCEDURE(PARM) OPTIONS(MAIN);

111:       'DIST  BT IM   LNGTH SRF RDY TYP  ADT    DHV   VOL  ' ||
112:       'END SRF DRN SAF CAP TOT ADJ  MILES';

113:    /*** READ TABLE OF CITY NAMES ***/
114:    OPEN FILE (TABLE) INPUT RECORD TITLE ('CITYTBL');
115:    DO J=1 TO #_CITIES;
116:       READ FILE (TABLE) SET (PTR_IN);
117:       CITY(J) = CITY_NAME;
118:       END;
119:    CLOSE FILE (TABLE);

120: /*** READ TABLE OF COUNTY NAMES ***/
121: OPEN FILE(TABLE) INPUT RECORD TITLE('CNTYTBL');
122: DO J = 1 TO #_COUNTIES;
123:       READ FILE(TABLE) SET(PTR_IN);
124:       COUNTY_TABLE(J) = COUNTY_NAME;
125:       END;
126:       COUNTY_TABLE(0) = '*** INVALID ***';
127: CLOSE FILE(TABLE);

128:    /*** INITIALIZE INPUT FILES ***/
129:    OPEN FILE(SUBSID) TITLE('SUFFSUB');
130:    OPEN FILE (ROADLOG) INPUT SEQL;
131:    ON ENDFILE (SUBSID) GOTO FINISH;
132:    ON KEY (SUBSID) BEGIN;
133:       PRINTER = '*** NO RECORD FOR STARTKEY ' || STARTKEY;
134:       CALL PRINTX (F(3));
135:       GOTO RETURN;
136:       END;
137:    ON KEY (ROADLOG) RLG.DESCR = '*** ROADLOG RECORD MISSING ***';
138:    READ FILE (SUBSID) SET (PTR_IN) KEY (STARTKEY);
139:    SAVE_RT_# = 0;
140:    J = CARRIAGE;
141:    SYS_SECTN, SYS_DEFIC = 0;


142: /**** MAIN EXECUTION LOOP *****/
143:    DO WHILE (IN1.KEY<=ENDKEY);

144:       /*** CHECK FOR NEW ROUTE ***/
145:       IF SAVE_RT_#¬=IN.RT_# THEN DO;
146:          IF SAVE_RT_#¬=0 THEN CALL TOTALS;
147:          ZRT_# = IN.RT_#;
148:          IF IN.SYSTEM='I' THEN HEADING(1) = SUBSTR(BLANKS,1,45) ||
149:             'FEDERAL AID INTERSTATE ROUTE NUMBER' || ZRT_#;
150:          ELSE IF IN.SYSTEM='P' THEN HEADING(1) = SUBSTR(BLANKS,1,45) ||
151:          'FEDERAL AID PRIMARY ROUTE NUMBER' || ZRT_#;
152:          ELSE HEADING(1) = SUBSTR(BLANKS,1,45) ||
153:             'FEDERAL AID SECONDARY ROUTE NUMBER ' || ZRT_#;
154:          IF SAVE_RT_#¬=0 & PAGE_SIZE-PAGE_POSITION>3 THEN DO;
155:             PRINTER = LINES;
156:             CALL PRINTX (F(3));
157:             END;
158:          IF PAGE_SIZE-PAGE_POSITION>10 THEN DO;
159:             PRINTER = HEADING(1);
160:             CALL PRINTX (F(3));
161:             J = 3;
162:             END;
```

```
SK: PROCEDURE(PARM) OPTIONS(MAIN);

163:          ELSE PAGE_POSITION = PAGE_SIZE;
164:          SAVE_RT_# = IN.RT_#;
165:          ROUTE_SECTN, ROUTE_DEFIC = 0;
166:          END;

167:       /*** CHECK FOR COINCIDENT SECTION ***/
168:       IF IN.REMARK='C' THEN DO;
169:          READ FILE (ROADLOG) KEY (IN1.KEY) SET (PTR_RLG);
170:          PRINTER = IN.MILEPOST || IN.FRACTION || SUBSTR(BLANKS,1,25) ||
171:          RLG.DESCR;
172:          CALL PRINTX (F(2));
173:          J = 2;
174:          GOTO NEXT;
175:          END;
176:       STRING_OUT = ' ';
177:       OUT.MIPOST = IN.MILEPOST || IN.FRACTION;
178:    OUT.COUNTY = COUNTY_TABLE(IN.CNTY);
179:       OUT.FINANCIAL_DIST = IN.FINAN;
180:    OUT.SECTN = IN.SECTION + .05;

181:       /*** CHECK FOR CITY ***/
182:       IF IN.REMARK='M' THEN DO;
183:       IF IN.CITY_# = 0
184:             THEN SUBSTR(STRING_OUT,70,21) = 'NON-INCORPORATED CITY';
185:             ELSE SUBSTR(STRING_OUT,70,26) = 'CITY OF ' ||
186:             CITY(IN.CITY_#);
187:          PRINTER = STRING_OUT;
188:          CALL PRINTX (F(2));
189:          J = 2;
190:          GOTO NEXT;
191:          END;
192:       OUT.DESCR = IN.DESCRIP;
193:       OUT.DEFIC_MLGE = IN.DEFIC;
194:       ROUTE_SECTN = ROUTE_SECTN + IN.SECTION;
195:       ROUTE_DEFIC = ROUTE_DEFIC + IN.DEFIC;

196:       /*** NON-EXISTENT OR UNDER CONSTRUCTION ***/
197:       IF IN.REMARK='N' | IN.REMARK='U' THEN DO;
198:          IF IN.REMARK='N'
199:             THEN SUBSTR(STRING_OUT,70,12) = 'NON EXISTENT';
200:             ELSE SUBSTR(STRING_OUT,70,18) = 'UNDER CONSTRUCTION';
201:          PRINTER = STRING_OUT;
202:          CALL PRINTX (F(J));
203:          J = CARRIAGE;
204:          GOTO NEXT;
205:          END;
206:    OUT = IN, BY NAME;
207:    OUT.ADT = IN.CURRENT_SECTION_ADT;
208:    PRINTER = STRING_OUT;
209:       CALL PRINTX (F(J));
210:       J = CARRIAGE;
211: NEXT: READ FILE(SUBSID) SET(PTR_IN);
212:       IF IN.DESCRIP = 'END OF ROUTE      ' THEN GO TO NEXT;
213:       END;

214: FINISH:
215:    CALL TOTALS;
216:    OUT.DESCR = '  SYSTEM TOTAL';
```

```
SK: PROCEDURE(PARM) OPTIONS(MAIN);

217:     OUT.SECTN = SYS_SECTN + .05;
218:     OUT.DEFIC_MLGE = SYS_DEFIC;
219:     OUT.ADJ_RATING = (SYS_DEFIC/SYS_SECTN)*100 + .5;
220:     PRINTER = STRING_OUT;
221:     #_HDGS = 2;
222:     CALL PRINTX (F(3));

223: RETURN:
224:     CLOSE FILE (SUBSID);
225:     CLOSE FILE (ROADLOG);
226:     CALL EXIT(PARM);
227:     RETURN;


228: /***** SUBROUTINE TO PRINT TOALS OF ROUTE *****/

229: TOTALS:  PROCEDURE;
230:     STRING_OUT = ' ';
231:     OUT.DESCR = '  ROUTE TOTAL';
232:     OUT.SECTN = ROUTE_SECTN + .05;
233:     OUT.DEFIC_MLGE = ROUTE_DEFIC;
234:     OUT.ADJ_RATING = (ROUTE_DEFIC / ROUTE_SECTN)*100 + .5;
235:     PRINTER = STRING_OUT;
236:     #_HDGS = 2;
237:     CALL PRINTX (F(3));
238:     #_HDGS = 5;
239:     SYS_DEFIC = SYS_DEFIC + ROUTE_DEFIC;
240:     SYS_SECTN = SYS_SECTN + ROUTE_SECTN;
241:     END TOTALS;

242: END SK;
```

LIST-BY-RATING --

Member Name	SRS	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- SRS output
	SUFFREP	-- A sorted copy of the Sufficiency Report file
	CNTYTBL	-- Table of county names
Instruction	1 - 3	"SRS"

LIST-BY-RATING provides a listing of the sufficiency sections in order of the adjusted rating for each section. The listing begins with those records having the lowest adjusted ratings. The SUFFREP File needed to execute this program is created by copying the Sufficiency Report file into the file SUFFREP through the program COPY-FOR-SORTING. Following the copy procedure the SUFFREP file is then sorted by adjusted rating. The sorting is done by the IBM SORT/MERGE Utility Program.

The SRS program listing follows:

```
SRS: PROCEDURE(PARM) OPTIONS(MAIN);

  1: SRS: PROCEDURE(PARM) OPTIONS(MAIN);


  2: /***** DECLARATION OF VARIABLES *****/

  3: DECLARE
  4:    #_CITIES DEC FIXED (3) INIT(126),
  5:    #_COUNTIES DEC FIXED (2) INIT(56),
  6:    #_HDGS PIC'Z' DEF INSTR POS(72),
  7:    BLANKS CHAR(100) INIT(' '),
  8:    CARRIAGE DEC FIXED (1) INIT (1),
  9:    CITY(126) CHAR(18),
 10:    CITY_NAME CHAR(18) BASED (PTR_IN),
 11:    COUNTY_TABLE(0:56) CHAR(15),
 12:    1 COUNTY_STRUCTURE BASED(PTR_IN),
 13:       3 DUMMY1 CHAR(15),
 14:       3 COUNTY_NAME CHAR(15),
 15:    ENDKEY CHAR(13) DEF INSTR POS(56),
 16:    F(0:9) PIC'Z' INIT(0,1,2,3,4,5,6,7,8,9),
 17:    HEADING(9) CHAR(132) EXT,
 18:    1  IN BASED (PTR_IN),
 19:       3  DUM1 CHAR(1),
 20:       3  SYSTEM CHAR(1),
 21:       3  RT_# PIC'ZZZ',
 22:       3  MILEPOST PIC'999',
 23:       3  FRACTION PIC'+9V.999',
 24:       3  REMARK CHAR(1),
 25:       3  DESCRIP CHAR(18),
 26:       3  CNTY PIC'ZZ',
 27:       3  FINAN PIC'ZZ',
 28:       3  YR_BLT PIC'ZZ',
 29:       3  YR_IMP PIC'ZZ',
 30:       3  SURF_WIDTH PIC'ZZ',
 31:       3  RDY_WIDTH PIC'ZZ',
 32:       3  SURF_TYPE CHAR(3),
 33:       3  SECTION PIC'ZZZVZZZ',
 34:       3  ADT PIC'ZZZZZ',
 35:       3  DHV PIC'ZZZZ',
 36:       3  PERCENT_TRUCKS PIC'ZZ',
 37:       3  SERVICE_VOL PIC'ZZZZ',
 38:       3  #_ACCIDENTS PIC'ZZ',
 39:       3  FND_RATING PIC'ZZ',
 40:       3  SRF_RATING PIC'ZZ',
 41:       3  DRN_RATING PIC'ZZ',
 42:       3  SAF_RATING PIC'ZZ',
 43:       3  CAP_RATING PIC'ZZ',
 44:       3  TOT_RATING PIC'ZZZ',
 45:       3  ADJ_RATING PIC'ZZZ',
 46:       3  DEFIC PIC'ZZVZZ',
 47:       3 DESIGN_SPEED PIC'ZZ',
 48:       3  TERRAIN CHAR(1),
 49:       3  AVG_SPEED PIC'ZZ',
 50:       3  SIGHT_DIST PIC'ZZ',
 51:       3 STOP_DIST PIC'VZZ',
 52:       3  CURVES PIC'ZZ',
 53:       3  BRIDGES PIC'Z',
 54:       3  #_LANES PIC'Z',
 55:       3  DIVIDED_CODE CHAR(1),
```

```
SRS: PROCEDURE(PARM) OPTIONS(MAIN);

 56:       3  CITY_# PIC'ZZZ',
 57:       3  CURRENT_SECTION_ADT PIC'ZZZZZ',
 58: 1 IN1 BASED (PTR_IN),
 59:       3  DUM CHAR(1),
 60:       3  KEY CHAR(13),
 61:    INSTR CHAR(80) EXT,
 62:    LINES CHAR(128) INIT((128)'-'),
 63:    1  OUT DEF STRING_OUT,
 64:       3  MIPOST CHAR(10),
 65:       3  DESCR CHAR(20),
 66:       3  COUNTY CHAR(16),
 67:       3  FINANCIAL_DIST PIC'ZZ',
 68:       3  YR_BLT PIC'ZZZZZ',
 69:       3  YR_IMP PIC'ZZZZ',
 70:       3  SECTN PIC'ZZZZZV.Z',
 71:       3  SURF_WIDTH PIC'ZZZZ',
 72:       3  RDY_WIDTH PIC'ZZZZB',
 73:       3  SURF_TYPE CHAR(3),
 74:       3  ADT PIC'ZZZZZZ',
 75:       3  DHV PIC'ZZZZZZ',
 76:       3  SERVICE_VOL PIC'ZZZZZZ',
 77:       3  FND_RATING PIC'ZZZZ9',
 78:       3  SRF_RATING PIC'ZZZ9',
 79:       3  DRN_RATING PIC'ZZZ9',
 80:       3  SAF_RATING PIC'ZZZ9',
 81:       3  CAP_RATING PIC'ZZZ9',
 82:       3  TOT_RATING PIC'ZZZ9',
 83:       3  ADJ_RATING PIC'ZZZ9',
 84:       3  DEFIC_MLGE PIC'ZZZZV.Z',
 85:    PAGE_POSITION PIC'ZZ' DEF INSTR POS(9),
 86:    PAGE_SIZE PIC'ZZ' DEF INSTR POS(7),
 87:    PARM                                  CHAR(100),
 88:    PRINTER CHAR(132) EXT,
 89:    PTR_IN PTR,
 90:    PTR_RLG PTR,
 91:    1  RLG BASED (PTR_RLG),
 92:       3 DUM CHAR(14),
 93:       3  DESCR CHAR(35),
 94:    ROADLOG FILE RECORD KEYED INPUT SEQL ENV(INDEXED),
 95:    ROUTE_DEFIC PIC'ZZZZVZ',
 96:    ROUTE_SECTN PIC'ZZZZVZZZ',
 97:    SAVE_RT_# PIC'ZZZ',
 98:    STARTKEY CHAR(13) DEF INSTR POS(40),
 99:    STRING_OUT CHAR(132),
100:    SUFFREP FILE RECORD,
101:    SYS_DEFIC PIC'ZZZZVZ',
102:    SYS_SECTN PIC'ZZZZVZZZ',
103:    ZRT_# PIC'ZZZBB';


104: /***** PROGRAM INITIALIZATION *****/
105:       CALL INIT(PARM);

106:    /*** SET UP COLUMN HEADINGS *****/
107:    #_HDGS = 5;
108:       HEADING(3) = '   MILE        SECTION                    ' ||
109:       '    FIN  *YEAR*  SECTN  WIDTH  SUF              SERV  ' ||
110:       '*** SUFFICIENCY RATINGS ***  DEFIC';
```

```
SRS: PROCEDURE(PARM) OPTIONS(MAIN);

111:       HEADING(4) = '   POST      DESCRIPTION                          ' ||
112:       'DIST  BT IM   LNGTH SRF RDY TYP  ADT    DHV    VOL   ' ||
113:       'END SRF DRN SAF CAP TOT ADJ  MILES';

114:    /*** READ TABLE OF CITY NAMES ***/
115:    OPEN FILE (TABLE) INPUT RECORD TITLE ('CITYTBL');
116:    DO J=1 TO #_CITIES;
117:       READ FILE (TABLE) SET (PTR_IN);
118:       CITY(J) = CITY_NAME;
119:       END;
120:    CLOSE FILE (TABLE);

121: /*** READ TABLE OF COUNTY NAMES ***/
122: OPEN FILE(TABLE) INPUT RECORD TITLE('CNTYTBL');
123: DO J = 1 TO #_COUNTIES;
124:       READ FILE(TABLE) SET(PTR_IN);
125:       COUNTY_TABLE(J) = COUNTY_NAME;
126:       END;
127:       COUNTY_TABLE(0) = '*** INVALID ***';
128: CLOSE FILE(TABLE);

129:    /*** INITIALIZE INPUT FILES ***/
130:    OPEN FILE(SUBSID) TITLE('SUFFREP');
131:    OPEN FILE (ROADLOG) INPUT SEQL;
132:    ON ENDFILE (SUBSID) GOTO FINISH;
133:    ON KEY (ROADLOG) RLG.DESCR = '*** ROADLOG RECORD MISSING ***';
134:    READ FILE(SUBSID) SET(PTR_IN);
135:    SAVE_RT_# = 0;
136:    J = CARRIAGE;
137:    SYS_SECTN, SYS_DEFIC = 0;


138: /**** MAIN EXECUTION LOOP *****/
139:    LOOP:

140:       /*** CHECK FOR NEW ROUTE ***/
141:    HEADING(1) = SUBSTR(BLANKS,1,45)||
142:         ' SUFFICIENCY RATINGS BY SECTION-STATEWIDE';

143:       /*** CHECK FOR COINCIDENT SECTION ***/
144:       STRING_OUT = ' ';
145:       OUT.MIPOST = IN.MILEPOST || IN.FRACTION;
146:    OUT.COUNTY = COUNTY_TABLE(IN.CNTY);
147:       OUT.FINANCIAL_DIST = IN.FINAN;
148:    OUT.SECTN = IN.SECTION + .05;
149:       OUT.DESCR = IN.DESCRIP;
150:       OUT.DEFIC_MLGE = IN.DEFIC;

151:       /*** NON-EXISTENT OR UNDER CONSTRUCTION ***/
152:       IF IN.REMARK='N' | IN.REMARK='U' THEN DO;
153:          IF IN.REMARK='N'
154:             THEN SUBSTR(STRING_OUT,70,12) = 'NON EXISTENT';
155:             ELSE SUBSTR(STRING_OUT,70,18) = 'UNDER CONSTRUCTION';
156:          PRINTER = STRING_OUT;
157:          CALL PRINTX (F(J));
158:          J = CARRIAGE;
159:          GOTO NEXT;
160:          END;
```

```
SRS: PROCEDURE(PARM) OPTIONS(MAIN);

161:    OUT = IN, BY NAME;
162:    OUT.ADT = IN.CURRENT_SECTION_ADT;
163:    PRINTER = STRING_OUT;
164:       CALL PRINTX (F(J));
165:       J = CARRIAGE;
166: NEXT: READ FILE(SUBSID) SET(PTR_IN);
167:    GO TO LOOP;

168: FINISH:
169:    OUT.DESCR = '  SYSTEM TOTAL';
170:    OUT.SECTN = SYS_SECTN + .05;
171:    OUT.DEFIC_MLGE = SYS_DEFIC;
172:    OUT.ADJ_RATING = (SYS_DEFIC/SYS_SECTN)*100 + .5;
173:    PRINTER = STRING_OUT;
174:    #_HDGS = 2;
175:    CALL PRINTX (F(3));

176: RETURN:
177:    CLOSE FILE (SUBSID);
178:    CLOSE FILE (ROADLOG);
179:    CALL EXIT(PARM);
180:    RETURN;
181:    END SRS;
```

LIST-BY-DISTRICT --

Member Name	SLS	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- SLS output
	SUFFREP	-- A sorted copy of the Sufficiency Report file
	CNTYTBL	-- Table of county names
Instruction	1 - 3	"SLS"

LIST-BY-DISTRICT provides a listing of the sufficiency sections within each of Montana's financial districts. The Suffrep file needed to execute this program is created by copying the Sufficiency Report file into the file Suffrep through use of the program COPY-FOR-SORTING. Following the copy procedure the Suffrep File is then sorted by financial district and adjusted rating. The sorting is done by the IBM SORT/MERGE Utility Program.

The SLS program listing follows:

```
SLS: PROCEDURE(PARM) OPTIONS(MAIN);

  1: SLS: PROCEDURE(PARM) OPTIONS(MAIN);


  2: /***** DECLARATION OF VARIABLES *****/

  3: DECLARE
  4:    #_CITIES DEC FIXED (3) INIT(126),
  5:    #_COUNTIES DEC FIXED (2) INIT(56),
  6:    #_HDGS PIC'Z' DEF INSTR POS(72),
  7:    BLANKS CHAR(100) INIT(' '),
  8:    CARRIAGE DEC FIXED (1) INIT (1),
  9:    CITY(126) CHAR(18),
 10:    CITY_NAME CHAR(18) BASED (PTR_IN),
 11:    COUNTY_TABLE(0:56) CHAR(15),
 12:    1 COUNTY_STRUCTURE BASED(PTR_IN),
 13:       3 DUMMY1 CHAR(15),
 14:       3 COUNTY_NAME CHAR(15),
 15:    ENDKEY CHAR(13) DEF INSTR POS(56),
 16:    F(0:9) PIC'Z' INIT(0,1,2,3,4,5,6,7,8,9),
 17:    HEADING(9) CHAR(132) EXT,
 18:    1  IN BASED (PTR_IN),
 19:       3  DUM1 CHAR(1),
 20:       3  SYSTEM CHAR(1),
 21:       3  RT_# PIC'ZZZ',
 22:       3  MILEPOST PIC'999',
 23:       3  FRACTION PIC'+9V.999',
 24:       3  REMARK CHAR(1),
 25:       3  DESCRIP CHAR(18),
 26:       3  CNTY PIC'ZZ',
 27:       3  FINAN PIC'ZZ',
 28:       3  YR_BLT PIC'ZZ',
 29:       3  YR_IMP PIC'ZZ',
 30:       3  SURF_WIDTH PIC'ZZ',
 31:       3  RDY_WIDTH PIC'ZZ',
 32:       3  SURF_TYPE CHAR(3),
 33:       3  SECTION PIC'ZZZVZZZ',
 34:       3  ADT PIC'ZZZZZ',
 35:       3  DHV PIC'ZZZZ',
 36:       3  PERCENT_TRUCKS PIC'ZZ',
 37:       3  SERVICE_VOL PIC'ZZZZ',
 38:       3  #_ACCIDENTS PIC'ZZ',
 39:       3  FND_RATING PIC'ZZ',
 40:       3  SRF_RATING PIC'ZZ',
 41:       3  DRN_RATING PIC'ZZ',
 42:       3  SAF_RATING PIC'ZZ',
 43:       3  CAP_RATING PIC'ZZ',
 44:       3  TOT_RATING PIC'ZZZ',
 45:       3  ADJ_RATING PIC'ZZZ',
 46:       3  DEFIC PIC'ZZVZZ',
 47:       3 DESIGN_SPEED PIC'ZZ',
 48:       3  TERRAIN CHAR(1),
 49:       3  AVG_SPEED PIC'ZZ',
 50:       3  SIGHT_DIST PIC'ZZ',
 51:       3 STOP_DIST PIC'VZZ',
 52:       3  CURVES PIC'ZZ',
 53:       3  BRIDGES PIC'Z',
 54:       3  #_LANES PIC'Z',
 55:       3  DIVIDED_CODE CHAR(1),
```

```
SLS: PROCEDURE(PARM) OPTIONS(MAIN);

 56:       3  CITY_# PIC'ZZZ',
 57:       3  CURRENT_SECTION_ADT PIC'ZZZZZ',
 58: 1 IN1 BASED (PTR_IN),
 59:       3  DUM CHAR(1),
 60:       3  KEY CHAR(13),
 61:    INSTR CHAR(80) EXT,
 62:    LINES CHAR(128) INIT((128)'-'),
 63:    1  OUT DEF STRING_OUT,
 64:       3  MIPOST CHAR(10),
 65:       3  DESCR CHAR(20),
 66:       3  COUNTY CHAR(16),
 67:       3  FINANCIAL_DIST PIC'ZZ',
 68:       3  YR_BLT PIC'ZZZZZ',
 69:       3  YR_IMP PIC'ZZZZ',
 70:       3  SECTN PIC'ZZZZZV.Z',
 71:       3  SURF_WIDTH PIC'ZZZZ',
 72:       3  RDY_WIDTH PIC'ZZZZB',
 73:       3  SURF_TYPE CHAR(3),
 74:       3  ADT PIC'ZZZZZZ',
 75:       3  DHV PIC'ZZZZZZ',
 76:       3  SERVICE_VOL PIC'ZZZZZZ',
 77:       3  FND_RATING PIC'ZZZZ9',
 78:       3  SRF_RATING PIC'ZZZ9',
 79:       3  DRN_RATING PIC'ZZZ9',
 80:       3  SAF_RATING PIC'ZZZ9',
 81:       3  CAP_RATING PIC'ZZZ9',
 82:       3  TOT_RATING PIC'ZZZ9',
 83:       3  ADJ_RATING PIC'ZZZ9',
 84:       3  DEFIC_MLGE PIC'ZZZZV.Z',
 85:    PAGE_POSITION PIC'ZZ' DEF INSTR POS(9),
 86:    PAGE_SIZE PIC'ZZ' DEF INSTR POS(7),
 87:    PARM                                CHAR(100),
 88:    PRINTER CHAR(132) EXT,
 89:    PTR_IN PTR,
 90:    PTR_RLG PTR,
 91:    1  RLG BASED (PTR_RLG),
 92:       3 DUM CHAR(14),
 93:       3  DESCR CHAR(35),
 94:    ROADLOG FILE RECORD KEYED INPUT SEQL ENV(INDEXED),
 95:    ROUTE_DEFIC PIC'ZZZZVZ',
 96:    ROUTE_SECTN PIC'ZZZZVZZZ',
 97:    SAVE_DIST_# PIC'ZZ',
 98:    STARTKEY CHAR(13) DEF INSTR POS(40),
 99:    STRING_OUT CHAR(132),
100:    SUBSID FILE RECORD,
101:    SYS_DEFIC PIC'ZZZZVZ',
102:    SYS_SECTN PIC'ZZZZVZZZ',
103:    ZDIST_#   PIC'ZZZBB';


104: /***** PROGRAM INITIALIZATION *****/
105:       CALL INIT(PARM);

106:    /*** SET UP COLUMN HEADINGS *****/
107:    #_HDGS = 5;
108:       HEADING(3) = '   MILE        SECTION                    ' ||
109:       '    FIN  *YEAR*  SECTN  WIDTH  SUF                SERV  ' ||
110:       '*** SUFFICIENCY RATINGS ***  DEFIC';
```

```
SLS: PROCEDURE(PARM) OPTIONS(MAIN);

111:       HEADING(4) = '   POST      DESCRIPTION                          ' ||
112:       'DIST  BT IM   LNGTH SRF RDY TYP ) ADT    DHV    VOL   ' ||
113:       'END SRF DRN SAF CAP TOT ADJ  MILES';

114:    /*** READ TABLE OF CITY NAMES ***/
115:    OPEN FILE (TABLE) INPUT RECORD TITLE ('CITYTBL');
116:    DO J=1 TO #_CITIES;
117:       READ FILE (TABLE) SET (PTR_IN);
118:       CITY(J) = CITY_NAME;
119:       END;
120:    CLOSE FILE (TABLE);

121: /*** READ TABLE OF COUNTY NAMES ***/
122: OPEN FILE(TABLE) INPUT RECORD TITLE('CNTYTBL');
123: DO J = 1 TO #_COUNTIES;
124:       READ FILE(TABLE) SET(PTR_IN);
125:       COUNTY_TABLE(J) = COUNTY_NAME;
126:       END;
127:       COUNTY_TABLE(0) = '*** INVALID ***';
128: CLOSE FILE(TABLE);

129:    /*** INITIALIZE INPUT FILES ***/
130:    OPEN FILE(SUBSID) TITLE('SUFFREP');
131:    OPEN FILE (ROADLOG) INPUT SEQL;
132:    ON ENDFILE (SUBSID) GOTO FINISH;
133:    ON KEY (ROADLOG) RLG.DESCR = '*** ROADLOG RECORD MISSING ***';
134:       READ FILE(SUBSID) SET(PTR_IN);
135:    SAVE_DIST_# = 0;
136:    J = CARRIAGE;
137:    SYS_SECTN, SYS_DEFIC = 0;


138: /**** MAIN EXECUTION LOOP *****/
139:    DO WHILE (IN1.KEY<=ENDKEY);
140: -     /*** CHECK FOR NEW FINANCIAL DISTRICT ***/
141:             IF IN.FINAN = 0 THEN DO;
142:                  PRINTER = 'SUFFICIENCY RECORD AT MILEPOST  '||
143:                  IN.SYSTEM||IN.RT_#||IN.MILEPOST||IN.FRACTION||
144:                  '  HAS NO FINANCIAL DISTRICT STORED';
145:                  CALL PRINTX(F(1));
146:                  GO TO NEXT;
147:                  END;
148:     IF SAVE_DIST_# ¬= IN.FINAN THEN DO;
149:          IF SAVE_DIST_# ¬= 0 THEN CALL TOTALS;
150:          ADIST_# = IN.FINAN;
151:          HEADING(1) = SUBSTR(BLANKS,1,45) ||
152:             '           FINANCIAL DISTRICT NUMBER ' || ZDIST_#;
153:          IF SAVE_RT_#¬=0 & PAGE_SIZE-PAGE_POSITION>3 THEN DO;
154:             PRINTER = LINES;
155:             CALL PRINTX (F(3));
156:             END;
157:          IF PAGE_SIZE-PAGE_POSITION>10 THEN DO;
158:             PRINTER = HEADING(1);
159:             CALL PRINTX (F(3));
160:             J = 3;
161:             END;
162:          ELSE PAGE_POSITION = PAGE_SIZE;
163:          SAVE_DIST_# = IN.FINAN;
```

```
SLS: PROCEDURE(PARM) OPTIONS(MAIN);

164:          ROUTE_SECTN, ROUTE_DEFIC = 0;
165:          END;
166:       STRING_OUT = ' ';
167:       OUT.MIPOST = IN.MILEPOST || IN.FRACTION;
168:    OUT.COUNTY = COUNTY_TABLE(IN.CNTY);
169:       OUT.FINANCIAL_DIST = IN.FINAN;
170:    OUT.SECTN = IN.SECTION + .05;
171:       OUT.DESCR = IN.DESCRIP;
172:       OUT.DEFIC_MLGE = IN.DEFIC;
173:       ROUTE_SECTN = ROUTE_SECTN + IN.SECTION;
174:       ROUTE_DEFIC = ROUTE_DEFIC + IN.DEFIC;

175:       /*** NON-EXISTENT OR UNDER CONSTRUCTION ***/
176:       IF IN.REMARK='N' | IN.REMARK='U' THEN DO;
177:          IF IN.REMARK='N'
178:             THEN SUBSTR(STRING_OUT,70,12) = 'NON EXISTENT';
179:             ELSE SUBSTR(STRING_OUT,70,18) = 'UNDER CONSTRUCTION';
180:          PRINTER = STRING_OUT;
181:          CALL PRINTX (F(J));
182:          J = CARRIAGE;
183:          GOTO NEXT;
184:          END;
185:    OUT = IN, BY NAME;
186:    OUT.ADT = IN.CURRENT_SECTION_ADT;
187:    PRINTER = STRING_OUT;
188:       CALL PRINTX (F(J));
189:       J = CARRIAGE;
190: NEXT: READ FILE(SUBSID) SET(PTR_IN);
191:       END;

192: FINISH:
193:    CALL TOTALS;
194:    OUT.DESCR = '  SYSTEM TOTAL';
195:    OUT.SECTN = SYS_SECTN + .05;
196:    OUT.DEFIC_MLGE = SYS_DEFIC;
197:    OUT.ADJ_RATING = (SYS_DEFIC/SYS_SECTN)*100 + .5;
198:    PRINTER = STRING_OUT;
199:    #_HDGS = 2;
200:    CALL PRINTX (F(3));

201: RETURN:
202:    CLOSE FILE (SUBSID);
203:    CLOSE FILE (ROADLOG);
204:    CALL EXIT(PARM);
205:    RETURN;


206: /***** SUBROUTINE TO PRINT TOALS OF ROUTE *****/

207: TOTALS:  PROCEDURE;
208:    STRING_OUT = ' ';
209:    OUT.DESCR = 'DISTRICT TOTAL';
210:    OUT.SECTN = ROUTE_SECTN + .05;
211:    OUT.DEFIC_MLGE = ROUTE_DEFIC;
212:    OUT.ADJ_RATING = (ROUTE_DEFIC / ROUTE_SECTN)*100 + .5;
213:    PRINTER = STRING_OUT;
214:    #_HDGS = 2;
215:    CALL PRINTX (F(3));
```

```
SLS: PROCEDURE(PARM) OPTIONS(MAIN);

216:     #_HDGS = 5;
217:     SYS_DEFIC = SYS_DEFIC + ROUTE_DEFIC;
218:     SYS_SECTN = SYS_SECTN + ROUTE_SECTN;
219:   END TOTALS;
220:   END SLS;
```

MAP-TABLES --

Member Name	SXS	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- SXS output
	SUFFSUB	-- Sufficiency Report file
Instruction	1 - 3	"SXS"
	40 - 43	Beginning route number
	56 - 59	Ending route number

The Sufficiency by Sections report contains mapped sections of the Federal Aid Highway System. MAP-TABLES prints the records from the Sufficiency Report file in a table format for printing along with the mapped sections of roadway.

The SXS program listing follows:

```
SX: PROCEDURE(PARM) OPTIONS(MAIN);

  1: SX: PROCEDURE(PARM) OPTIONS(MAIN);

  2: /* FILE DECLARATIONS */

  3: DECLARE SUFFSUB FILE RECORD KEYED ENV(INDEXED);

  4: /* VARIABLE DECLARATIONS */

  5: DECLARE
  6:     CHAR100                          CHAR(100),
  7:     ENDKEY                           CHAR(13) DEF INSTR POS(56),
  8:     F(0:9) STATIC                    PIC'Z'
  9:                                      INIT(0,1,2,3,4,5,6,7,8,9),
 10:     HEADING(9)                       CHAR(132) EXT,
 11:     INSTR                            CHAR(80) EXT,
 12:    #_HDGS                            PIC'Z' DEF INSTR POS(72),
 13:     LINE_TITLE(18)                   CHAR(24),
 14:     OUTPUT_MILEPOST                  PIC'BZZ9V.9',
 15:     OUTPUT_SECTION_LENGTH            PIC'BZZ9V.9',
 16:     OUTPUT_DEFICIENT_MILEAGE         PIC'BZZV.ZZ',
 17:     PAGE_POSITION                    PIC'ZZ' DEF INSTR POS(7),
 18:     PAGE_SIZE                        PIC'ZZ' DEF INSTR POS(9),
 19:     PARM                             CHAR(100),
 20:     PRINTER                          CHAR(132) EXT,
 21:     PTR                              PTR,
 22:     STARTKEY                         CHAR(13) DEF INSTR POS(40),
 23:     STRING_TABLE                     CHAR(108),

 24:     1 SUBSID BASED(PTR),
 25:         5 DUMMY1                     CHAR(1),
 26:         5 KEY,
 27:             10 SYSTEM                CHAR(1),
 28:             10 ROUTE_#               PIC'ZZZ',
 29:             10 MILEPOST              PIC'ZZZ',
 30:             10 FRACTION              PIC'+ZV.ZZZ',
 31:         5 REMARK                     CHAR(1),
 32:         5 DESCR                      CHAR(18),
 33:         5 DUMMY3                     CHAR(4),
 34:         5 YR_BLT                     PIC'ZZ',
 35:         5 YR_IMP                     PIC'ZZ',
 36:         5 SUR_WD                     PIC'ZZ',
 37:         5 RDY_WD                     PIC'ZZ',
 38:         5 SURTYP                     PIC'ZZZ',
 39:         5 SECTION_LENGTH             PIC'ZZZVZZZ',
 40:         5 ADT                        PIC'ZZZZZ',
 41:         5 DHV                        PIC'ZZZZ',
 42:         5 PERCENT_TRUCKS             PIC'ZZ',
 43:         5 SERVICE_VOL                PIC'ZZZZ',
 44:         5 #_ACCIDENTS                PIC'ZZ',
 45:         5 FND_RATING                 PIC'ZZ',
 46:         5 SRF_RATING                 PIC'ZZ',
 47:         5 DRAN_RATING                PIC'ZZ',
 48:         5 SAFTEY_RATING              PIC'ZZ',
 49:         5 CAPACITY_RATING            PIC'ZZ',
 50:         5 SUFF_RATING                PIC'ZZZ',
```

```
SX: PROCEDURE(PARM) OPTIONS(MAIN);

51:          5 ADJ_RATING                    PIC'ZZZ',
52:          5 DEFICIENT_MILEAGE             PIC'ZZVZZ',

53:     TABLE(18,17)                         CHAR(6);


54: ON ERROR BEGIN;
55:     PRINTER = '*** TERMINAL ERROR IN PHASE SX ***';
56:     CALL PRINTX(F(1));
57:     GO TO CLOSE;
58:     END;


59:     CALL INIT(PARM);
60: ON ENDFILE(SUFFSUB) BEGIN;
61:     ENDKEY = '          ';
62:     GO TO PRINT;
63:     END;


64: /* INITIALIZE THE TABLE TITLES */

65: LINE_TITLE(1)  = 'BEGINNING MILE POST      ';
66: LINE_TITLE(2)  = 'LENGTH                   ';
67: LINE_TITLE(3)  = 'YEAR BUILT              ';
68: LINE_TITLE(4)  = 'YEAR IMPROVED           ';
69: LINE_TITLE(5)  = 'SURFACE WIDTH           ';
70: LINE_TITLE(6)  = 'ROADWAY WIDTH           ';
71: LINE_TITLE(7)  = 'SURFACE TYPE            ';
72: LINE_TITLE(8)  = 'ADT                     ';
73: LINE_TITLE(9)  = 'DHV                     ';
74: LINE_TITLE(10) = 'SERVICE_VOLUME          ';
75: LINE_TITLE(11) = 'FOUNDATION - 10         ';
76: LINE_TITLE(12) = 'SURFACE - 30            ';
77: LINE_TITLE(13) = 'DRAINAGE - 10           ';
78: LINE_TITLE(14) = 'SAFTEY - 20             ';
79: LINE_TITLE(15) = 'CAPACITY RATING - 30    ';
80: LINE_TITLE(16) = 'SUFFICIENCY RATING      ';
81: LINE_TITLE(17) = 'ADJUCTED RATING         ';
82: LINE_TITLE(18) = 'DEFICIENT MILEAGE       ';


83: /* READ SUFFICIENCY SUBSIDIARY FILE - SUFSUM */

84: TABLE = ' ';
85: READ FILE(SUFFSUB) SET(PTR) KEY(STARTKEY);
86: J = 1;
87: GO TO ALLOCATE;

88: INITIAL: TABLE = ' ';

89: BEGIN: READ FILE(SUFFSUB) SET(PTR);
90: IF ENDKEY < STRING(SUBSID.KEY) THEN GO TO PRINT;
91: IF SUBSID.DESCR = 'END OF ROUTE      ' THEN GO TO BEGIN;


92: /* ALLOCATE THE INPUT INFORMATION TO THE OUTPUT ARRAY */
```

```
SX: PROCEDURE(PARM) OPTIONS(MAIN);

 93: ALLOCATE: OUTPUT_MILEPOST = SUBSID.MILEPOST + SUBSID.FRACTION;
 94: OUTPUT_SECTION_LENGTH = SUBSID.SECTION_LENGTH;
 95: OUTPUT_DEFICIENT_MILEAGE = SUBSID.DEFICIENT_MILEAGE;
 96: STRING_TABLE = OUTPUT_MILEPOST||OUTPUT_SECTION_LENGTH||'    '||
 97:     YR_BLT||'    '||YR_IMP||'    '||SUR_WD||'    '||RDY_WD||'   '||
 98:     SURTYP||' '||ADT||'  '||DHV||'  '||SERVICE_VOL||'    '||FND_RATING
 99:     ||'    '||SRF_RATING||'    '||DRAN_RATING||'    '||SAFTEY_RATING||
100:     '    '||CAPACITY_RATING||'   '||SUFF_RATING||'   '||ADJ_RATING||
101:     OUTPUT_DEFICIENT_MILEAGE;

102: CHECK:IF SUBSID.REMARK='C' THEN STRING_TABLE=OUTPUT_MILEPOST||
103:    OUTPUT_SECTION_LENGTH||'    C     O     I     N     C     I'
104:    ||'     D     E     N     T';


105: /* CHECK FOR AN URBAN SECTION */

106: IF SUBSID.REMARK='M' THEN STRING_TABLE=OUTPUT_MILEPOST||
107:    OUTPUT_SECTION_LENGTH||'    U     R     B     A     N';


108: /* CHECK FOR AN UNDER CONSTRUCTION SECTION */

109: IF SUBSID.REMARK='U' THEN STRING_TABLE=OUTPUT_MILEPOST||
110:    OUTPUT_SECTION_LENGTH||'  U     N     DC    EO    RN     S'
111:    ||'     T     R     U     C     T     I     O     N';


112: /* CHECK FOR A NON EXSISTANT SECTION */

113: IF SUBSID.REMARK='N' THEN STRING_TABLE=OUTPUT_MILEPOST||
114:    OUTPUT_SECTION_LENGTH||'  N     O     N     E     X'
115:    ||'     I     S     T     E     N     T';

116: K=1;
117: DO I = 1 TO 18;
118:     TABLE(I,J) = SUBSTR(STRING_TABLE,K,6);
119:     K = (6*I) + 1;
120:     END;
121: J = J + 1;
122: IF J>= 18 THEN GO TO PRINT;
123: ELSE GO TO BEGIN;


124: /* PRINT THE TABLE */
125:3PRINT: M = 9;
126: DO I = 1 TO 18;
127:     PRINTER = LINE_TITLE(I)||STRING(TABLE(I,*));
128:     CALL PRINTX(F(M));
129:     M = 1;
130:     END;
131: J = 1;
132: IF ENDKEY < STRING(SUBSID.KEY) THEN GO TO CLOSE;
133: GO TO INITIAL;


134: /* CLOSE FILES */
```

```
SX: PROCEDURE(PARM) OPTIONS(MAIN);


135: CLOSE: CLOSE FILE(SUFFSUB);
136: CALL EXIT(PARM);
137: END SX;
```

DEF-MILES-BY-COUNTY --

Member Name	SWS	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- SWS output
	SUFFSUB	-- Sufficiency Report file
	CNTYTBL	-- Table of county names
Instruction	1 - 3	"SWS"

DEF-MILES-BY-COUNTY summarizes the amount of deficient Federal Aid Mileage within each county. The amount of deficient mileage within each state financial district is also tabulated and printed out.

The SWS program listing follows:

```
SY: PROCEDURE(PARM) OPTIONS(MAIN);

 1: SY: PROCEDURE(PARM) OPTIONS(MAIN);

 2: /* FILE DECLARATIONS */

 3: DECLARE SUFFSUB FILE RECORD KEYED ENV(INDEXED);

 4: /* VARIABLE DECLARATIIONS */

 5: DECLARE
 6:     CHAR100                          CHAR(100),
 7:     ENDKEY                           CHAR(13) DEF INSTR POS(56),
 8:     F(0:9) STATIC                    PIC'Z'
 9:                                      INIT(0,1,2,3,4,5,6,7,8,9),
10:     HEADING(9)                       CHAR(132) EXT,
11:     INSTR                            CHAR(80) EXT,
12:     #_HDGS                           PIC'Z' DEF INSTR POS(72),
13:     MILES_ARRAY(14,11)               PIC'BZZZZZV.Z',
14:     PERCENT                          PIC'Z9V.9',
15:     PERCENT_ARRAY(14,11)             PIC'BBBZZZV.Z',
16:     PARM                             CHAR(100),
17:     PRINTER                          CHAR(132) EXT,
18:     PTR                              PTR,
19:     ROW_TITLE(11)                    CHAR(20),
20:     STARTKEY                         CHAR(13) DEF INSTR POS(40),

21:     1 SUBSID BASED(PTR),
22:         5 DUMMY1                     CHAR(1),
23:         5 KEY                        CHAR(13),
24:         5 REMARK                     CHAR(1),
25:         5 DESCR                      CHAR(18),
26:         5 DUMMY3                     CHAR(2),
27:         5 FINANCIAL_DISTRICT         PIC'ZZ',
28:         5 DUMMY4                     CHAR(11),
29:         5 SECTION_LENGTH             PIC'ZZZVZZZ',
30:         5 DUMMY5                     CHAR(30),
31:         5 ADJ_RATING                 PIC'ZZ9',
32:         5 DEFICIENT_MILEAGE          PIC'ZZVZZ';

33: ON ERROR BEGIN;
34:     PRINTER = '*** TERMINAL ERROR IN PHASE SY **';
35:     CALL PRINTX(F(1));
36:     GO TO CLOSE;
37:     END;

38: CALL INIT(PARM);

39: ON ENDFILE(SUFFSUB) GO TO ALLOCATE_2;

40: /* SET THE HEADINGS FOR THE TABLE */
```

```
SY: PROCEDURE(PARM) OPTIONS(MAIN);


41: #_HDGS = 6;
42: HEADING(2) = '
43:     ||'MILES OF RURAL HIGHWAY'||
44:     '                                                    ';
45: HEADING(3) = '                                                    '
46:     ||'DISTRIBUTED BY ADJUSTED SUFFICIENCY RATING'||
47:     '                                                    ';
48: HEADING(4)='
49:     ||'IN TEN PERCENT INCREMENTS'||
50:     '                                                    ';


51: /* READ SUFFICIENT SUBSIDIARY FILE - SUFSUM */

52: MILES_ARRAY = 0;
53: PERCENT_ARRAY = 0;


54: READ FILE(SUFFSUB) SET(PTR) KEY(STARTKEY);
55: IF SUBSID.REMARK = 'M' | SUBSID.REMARK = 'C'
56:     | SUBSID.DESCR = 'END OF ROUTE      ' THEN GO TO BEGIN;
57: GO TO ALLOCATE;


58: BEGIN: READ FILE(SUFFSUB) SET(PTR);
59: IF SUBSID.REMARK = 'M' | SUBSID.REMARK = 'C'
60:     | SUBSID.DESCR = 'END OF ROUTE      ' THEN GO TO BEGIN;


61: /* ALLOCATE SECTION LENGTH TO THE MILES ARRAY */

62: ALLOCATE: J = FINANCIAL_DISTRICT;
63: IF J=0 THEN J=12;
64: IF SUBSID.REMARK = 'U' THEN ADJ_RATING = 95;
65: IF ADJ_RATING <= 10 THEN M = 1;
66:     ELSE IF ADJ_RATING <= 20 THEN M = 2;
67:     ELSE IF ADJ_RATING <= 30 THEN M = 3;
68:     ELSE IF ADJ_RATING <= 40 THEN M = 4;
69:     ELSE IF ADJ_RATING <= 50 THEN M = 5;
70:     ELSE IF ADJ_RATING <= 60 THEN M = 6;
71:     ELSE IF ADJ_RATING <= 70 THEN M = 7;
72:     ELSE IF ADJ_RATING <= 80 THEN M = 8;
73:     ELSE IF ADJ_RATING <= 90 THEN M = 9;
74:     ELSE M = 10;

75: MILES_ARRAY(J,M) = MILES_ARRAY(J,M) + SECTION_LENGTH;
76: IF ENDKEY < SUBSID.KEY THEN GO TO ALLOCATE_2;
77: GO TO BEGIN;


78: /* DETERMINE WHAT PERCENT EACH MILE ARRAY ELEMENT IS OF
79:     ITS RESPECTIVE FINANCIAL DISTRICT */

80: ALLOCATE_2: DO I = 1 TO 12;
81:     DO J = 1 TO 10;
82:         MILES_ARRAY(I,11) = MILES_ARRAY(I,11) + MILES_ARRAY(I,J);
```

```
SY: PROCEDURE(PARM) OPTIONS(MAIN);

 83:     END;
 84: END;

 85: DO I = 1 TO 12;
 86:     DO J = 1 TO 10;
 87:     IF MILES_ARRAY(I,11) = 0 THEN PERCENT_ARRAY(I,J) =0;
 88:     ELSE PERCENT_ARRAY(I,J) = (MILES_ARRAY(I,J)/MILES_ARRAY(I,11))*100;
 89:     END;
 90: END;


 91: /* CALCULATE TOTAL MILEAGES & PERCENTS FOR THE SYSTEM */

 92: DO I = 1 TO 10;
 93:     DO J = 1 TO 12;
 94:         MILES_ARRAY(13,I) = MILES_ARRAY(13,I)+MILES_ARRAY(J,I);
 95:     END;
 96: MILES_ARRAY(13,11) = MILES_ARRAY(13,11) + MILES_ARRAY(13,I);
 97: END;

 98: DO I = 1 TO 10;
 99: PERCENT_ARRAY(13,I) = (MILES_ARRAY(13,I)/ MILES_ARRAY(13,11))*100.;
100: END;


101: /* ACCUMULATED TOTALS & PERCENTS */

102: MILES_ARRAY(14,1) = MILES_ARRAY(13,1);
103: PERCENT_ARRAY(14,1) = (MILES_ARRAY(13,1)/MILES_ARRAY(13,11))*100.;

104: DO I = 2 TO 10;
105:     J = I -1;
106:     MILES_ARRAY(14,I) = MILES_ARRAY(14,J) + MILES_ARRAY(13,I);
107:     PERCENT_ARRAY(14,I)= (MILES_ARRAY(14,I)/MILES_ARRAY(13,11))*100.;
108: END;

109: PERCENT_ARRAY(*,11) = 100.0;
110: PERCENT_ARRAY(14,10) = 100.0;
111: PERCENT_ARRAY(14,11) = 0;


112: /* PRINT OUT THE SUMMARY */

113: PRINTER='                                                    '||
114:         'FINANCIAL DISTRICTS'||
115:         '                                                    ';
116: CALL PRINTX(F(1));

117: PRINTER = ' PERCENT  '||
118:         '                                                    '||
119:         '                                                    '||
120:         '      ACCUM.';
121: CALL PRINTX(F(1));

122: PRINTER = 'SUFFICIENT          '||
123:        '    1       2       3       4       5       6       7    '||
124:        '   8       9       10      11      12    TOTALS  TOTALS';
125: CALL PRINTX(F(1));
```

```
SY: PROCEDURE(PARM) OPTIONS(MAIN);


126: ROW_TITLE(1)='   0-10      MILES   ';
127: ROW_TITLE(2)='  11-20      MILES   ';
128: ROW_TITLE(3)='  21-30      MILES   ';
129: ROW_TITLE(4)='  31-40      MILES   ';
130: ROW_TITLE(5)='  41-50      MILES   ';
131: ROW_TITLE(6)='  51-60      MILES   ';
132: ROW_TITLE(7)='  61-70      MILES   ';
133: ROW_TITLE(8)='  71-80      MILES   ';
134: ROW_TITLE(9)='  81-90      MILES   ';
135: ROW_TITLE(10)=' 91-100      MILES   ';
136: ROW_TITLE(11)='  TOTAL      MILES   ';

137: DO I = 1 TO 11;
138: PRINTER = ROW_TITLE(I)||STRING(MILES_ARRAY(*,I));
139: CALL PRINTX(F(2));
140: PRINTER = ' PERCENT       %      '||STRING(PERCENT_ARRAY(*,I));
141: CALL PRINTX(F(1));
142: END;


143: /* CLOSE FILES */

144: CLOSE: CLOSE FILE(SUFFSUB);
145: CALL EXIT(PARM);
146: END SY;
```

RATING-BY-DISTRICT --

Member Name	SZS	
Language	PL/I	
Subroutine	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- SZS output
	SUFFSUB	-- Sufficiency Report file
Instruction	1 - 3	"SZS"

RATING-BY-DISTRICT tabulates the miles of rural highway in each financial district by adjusted sufficiency ratings. The percentages of deficient mileages are printed along with the miles tabulated for the financial districts.

The SZS program listing follows:

```
SZ: PROCEDURE(PARM) OPTIONS(MAIN);

  1: SZ: PROCEDURE(PARM) OPTIONS(MAIN);


  2: /* FILE DECLARATIONS */

  3: DECLARE CNTYTBL FILE RECORD;

  4: DECLARE SUFFSUB FILE RECORD KEYED ENV(INDEXED);


  5: /* VARIABLE DECLARATION */

  6: DECLARE
  7:      1 CHAR80 BASED(CNTY_PTR),
  8:           5 COUNTY_NAME                  CHAR(15),
  9:           5 DUMMY1                       CHAR(34),
 10:           5 CNTY_FINANCIAL_DIST          PIC'ZZ',
 11:           5 DUMMY2                       CHAR(29),
 12:     COUNTY_MILEAGE(56)                   PIC'BBZZZZ9V.9BB',
 13:      CNTY_PTR                            PTR,
 14:     DEFICIENT_MILES(56)                  PIC'BBZZZZ9V.9BB',
 15:     ENDKEY                               CHAR(13) DEF INSTR POS(56),
 16:      F(0:9) STATIC                       PIC'Z'
 17:                                          INIT(0,1,2,3,4,5,6,7,8,9),
 18:      HDG(8,2)                            CHAR(10),
 19:      HEADING(9)                          CHAR(132) EXT,
 20:      INSTR                               CHAR(80) EXT,
 21:      #_HDGS                              PIC'Z' DEF INSTR POS(72),
 22:      PARM                                CHAR(100),
 23:      PERCENT                             PIC'Z9V.9',
 24:      PRINTER                             CHAR(132) EXT,
 25:      PTR                                 PTR,
 26:     STARTKEY                             CHAR(13) DEF INSTR POS(40),

 27:      1 SUBSID BASED(PTR),
 28:           5 DUMMY1                       CHAR(1),
 29:          5 KEY                           CHAR(13),
 30:           5 REMARK                       CHAR(1),
 31:           5 DESCRIPTION                  CHAR(18),
 32:           5 COUNTY_#                     PIC'ZZ',
 33:           5 FINANCIAL_DISTRICT           PIC'ZZ',
 34:           5 DUMMY2                       CHAR(11),
 35:           5 SECTION_LENGTH               PIC'ZZZVZZZ',
 36:           5 DUMMY3                       CHAR(33),
 37:           5 DEFICIENT_MILEAGE            PIC'ZZVZZ',
 38:      TOTAL_FIN_MILES                     PIC'BBZZZZZV.ZBB',
 39:      TOTAL_FIN_DEF                       PIC'BBZZZZZV.ZBB',
 40:      TOTAL_MILEAGE                       PIC'BBZZZZZV.ZBB',
 41:      TOTAL_DEFICIENT                     PIC'BBZZZZZV.ZBB',
 42:      TOTAL_SYS_MILES                     PIC'BBZZZZZV.ZBB',
 43:      TOTAL_SYS_DEF                       PIC'BBZZZZZV.ZBB';


 44: ON ERROR BEGIN;
 45:      PRINTER = '*** TERMINAL ERROR IN PHASE SZ ***';
 46:      CALL PRINTX(F(1));
 47:      GO TO CLOSE;
 48:      END;
```

```
SZ: PROCEDURE(PARM) OPTIONS(MAIN);


 49: CALL INIT(PARM);


 50: ON ENDFILE(SUFFSUB) GO TO PRINT;


 51: /* OPEN FILES */

 52: OPEN FILE(SUFFSUB) INPUT SEQL;


 53: /* TABLE HEADINGS */
 54:     #_HDGS = 6;
 55:    HEADING(1) = ((36)' ')||'RURAL DEFICIENT MILEAGE BY FINANCIAL'||
 56:          ' DISTRICTS AND PERCENTAGES';
 57:    HEADING(2) = ((41)' ')||'BASED ON SUFFICIENCY RATING FORMULA'||
 58:          ' PRIMARY SYSTEM';
 59:    HEADING(4) = ((36)' ')||'FINANCIAL                    RURAL   '||
 60:          'DEF. RURAL   PERCENT';
 61:    HEADING(5) = ((36)' ')||'DISTRICTS     COUNTY        MILEAGE  '||
 62:          '  MILEAGE   DEFICIENT';


 63: /* INITIALIZATION */
 64:     TOTAL_MILEAGE = 0;
 65:     TOTAL_DEFICIENT = 0;
 66:     COUNTY_MILEAGE = 0;
 67:     DEFICIENT_MILES =0;
 68:     TOTAL_SYS_MILES = 0;
 69:     TOTAL_SYS_DEF = 0;


 70: /* READ THE SUFFICIENCY SUBSIDIARY FILE */
 71: READ FILE(SUFFSUB) SET(PTR) KEY(STARTKEY);
 72: IF SUBSID.COUNTY_# = 0 THEN GO TO BEGIN;
 73: IF SUBSID.REMARK ¬= 'M' | SUBSID.REMARK ¬= 'N' | SUBSID.REMARK ¬= 'C'
 74:     THEN GO TO ALLOCATE;
 75: BEGIN: READ FILE(SUFFSUB) SET(PTR);
 76: IF SUBSID.REMARK = 'M' | SUBSID.REMARK = 'N' | SUBSID.REMARK = 'C'
 77:     THEN GO TO BEGIN;
 78: IF SUBSID.COUNTY_# = 0 THEN GO TO BEGIN;


 79: /* ALLOCATE DEFICIENT AND SECTION MILEAGES BY COUNTY */

 80: ALLOCATE: COUNTY_MILEAGE(COUNTY_#) = COUNTY_MILEAGE(COUNTY_#) +
 81:     SECTION_LENGTH;
 82: DEFICIENT_MILES(COUNTY_#) = DEFICIENT_MILES(COUNTY_#) +
 83:     DEFICIENT_MILEAGE;
 84: TOTAL_MILEAGE = TOTAL_MILEAGE + SECTION_LENGTH;
 85: TOTAL_DEFICIENT = TOTAL_DEFICIENT + DEFICIENT_MILEAGE;
 86: GO TO BEGIN;


 87: /* PRINT OUT THE SUMMARY */
```

```
SZ: PROCEDURE(PARM) OPTIONS(MAIN);


 88: PRINT: DO I = 1 TO 12;
 89:    IF I = 8 THEN K=9;
 90:    ELSE K=2;
 91:     TOTAL_FIN_MILES = 0;
 92:     TOTAL_FIN_DEF = 0;
 93:     OPEN FILE(CNTYTBL) INPUT SEQL;
 94:     DO J = 1 TO 56;
 95:          READ FILE(CNTYTBL) SET(CNTY_PTR);
 96:          IF CNTY_FINANCIAL_DIST = I THEN DO;
 97:          TOTAL_FIN_MILES = TOTAL_FIN_MILES + COUNTY_MILEAGE(J);
 98:          TOTAL_FIN_DEF = TOTAL_FIN_DEF + DEFICIENT_MILES(J);
 99:          IF COUNTY_MILEAGE(J) = 0 THEN PERCENT = 0;
100:          ELSE PERCENT=(DEFICIENT_MILES(J)/COUNTY_MILEAGE(J))*100;
101:         PRINTER = ((46)' ')||COUNTY_NAME||COUNTY_MILEAGE(J)||
102:               DEFICIENT_MILES(J)||'      '||PERCENT;
103:          CALL PRINTX(F(K));
104:          K=1;
105:          END;
106:     END;
107:     CLOSE FILE(CNTYTBL);
108:     IF TOTAL_FIN_MILES = 0 THEN PERCENT = 0;
109:          ELSE PERCENT = (TOTAL_FIN_DEF/TOTAL_FIN_MILES)*100;
110:     FINANCIAL_DISTRICT = I;
111:     PRINTER = ((40)' ')||FINANCIAL_DISTRICT||'                    '||
112:          TOTAL_FIN_MILES|| TOTAL_FIN_DEF || '      '||PERCENT;
113: CALL PRINTX(F(1));
114: TOTAL_SYS_MILES = TOTAL_SYS_MILES + TOTAL_FIN_MILES;
115: TOTAL_SYS_DEF = TOTAL_SYS_DEF + TOTAL_FIN_DEF;
116: END;
117: PERCENT = (TOTAL_SYS_DEF / TOTAL_SYS_MILES)*100;
118: PRINTER = ((36)' ')||' TOTAL   '||'                '||TOTAL_SYS_MILES|
119:     TOTAL_SYS_DEF||'      '||PERCENT;
120: CALL PRINTX(F(2));


121: /* CLOSE FILES */
122: CLOSE: CLOSE FILE(SUFFSUB);
123: CALL EXIT(PARM);
124: END;
```

COPY --

Member Name	PBS	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- PBS output
	SUFFICY	-- Sufficiency file
	SAVESUF	-- Backup copy (output)
Instruction	1 - 3	"PBS"
	5	"Y"/"N" for LIST=YES/LIST=NO

COPY prepares a backup copy of the Sufficiency file. The backup copy is a sequential version of the Sufficiency file, with identical record length (64 characters). A dummy record containing the date is first written. This record is followed by the Sufficiency records. If LIST=YES is specified, the records are listed in the same format as the Sufficiency file is stored. A count of the number of records copied into the backup file is taken. The count is printed after the last record is written.

The PBS program listing follows:

```
/* :COPY,FILE=SUFFICY,LIST=YES/NO */

 1: /* :COPY,FILE=SUFFICY,LIST=YES/NO */

 2: COPY:  PROCEDURE (PARM) OPTIONS (MAIN);

 3: /* INSTRUCTION */
 4: DECLARE
 5:    INSTR CHAR(80) EXT,
 6:    LIST CHAR(1) DEF INSTR POS(5),
 7:    #_HDGS PIC'Z' DEF INSTR POS(72);

 8: /* PRINT ROUTINE */
 9: DECLARE
10:    PARM CHAR(100),
11:   (HEADING(9),PRINTER) CHAR(132) EXT,
12:    PRINTX ENTRY (PIC'Z');

13: /* FILES */
14: DECLARE
15:    RECORD CHAR(56) BASED (PTR),
16:    BACKDD CHAR(8) STATIC INIT ('SAVESUF'),
17:    PERMDD CHAR(8) STATIC INIT ('SUFFICY'),
18:    PERM FILE RECORD KEYED ENV (INDEXED),
19:    BACKUP FILE RECORD;

20: /* OTHER VARIABLES */
21: DECLARE
22:    UD CHAR(6),
23:    CNTR BIN FIXED (31),
24:    PCNTR PIC'ZZZZZ9';


25: /***** INITIALIZATION *****/

26:    CALL INIT (PARM);

27:    /* SET UP HEADINGS */
28:    #_HDGS = 2;
29:    HEADING(1) = PERMDD || 'FILE COPY ROUTINE';

30:    /* INIT FILES */
31:    OPEN FILE (PERM) INPUT TITLE (PERMDD);
32:    OPEN FILE (BACKUP) OUTPUT TITLE (BACKDD);
33:    ON ENDFILE (PERM) GOTO DONE;

34:    /* RECORD DATE */
35:    UD = DATE;
36:    PTR = ADDR(HEADING(9));
37:    RECORD = SUBSTR(UD,3,2) || '/' ||
38:         SUBSTR(UD,5,2) || '/' || SUBSTR(UD,1,2);
39:    WRITE FILE (BACKUP) FROM (RECORD);


40: /***** MAIN EXECUTION LOOP *****/

41:    DO CNTR=1 TO 999999;
42:       READ FILE (PERM) SET (PTR);
43:       WRITE FILE (BACKUP) FROM (RECORD);
44:       IF LIST='Y' THEN DO;
```

```
/* :COPY,FILE=SUFFICY,LIST=YES/NO */

 45:           PRINTER = '      ' || RECORD;
 46:           CALL PRINTX (1);
 47:           END;
 48:       END;

 49: DONE:
 50:     PCNTR = CNTR - 1;
 51:     PRINTER = 'NUMBER OF RECORDS IN FILE:  ' || PCNTR;
 52:     CALL PRINTX (3);
 53:     CLOSE FILE (PERM);
 54:     CLOSE FILE (BACKUP);
 55:     CALL EXIT (PARM);

 56: END COPY;
```

CREATE --

Member Name	PAS	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- PAS output
	SUFFICY	-- Sufficiency file
	SAVESUF	-- Backup copy
Instruction	1 - 3	"PAS"
	5	"Y"/"N" for LIST=YES/LIST=NO

CREATE restores the Sufficiency file from a backup copy saved via program COPY. The first record in the file is a dummy record, containing the date on which the file was copied. This date is printed prior to performing the create operation. After printing the date, the records are read from the backup copy and written into the Sufficiency Data File, destroying the previous file. If LIST=YES is specified, the Sufficiency file is printed as the file is created. The records are counted as they are written. The count is printed after the create operation is complete.

The PAS program listing follows:

```
/* :CREATE,FILE=SUFFICY,LIST=YES/NO */

 1: /* :CREATE,FILE=SUFFICY,LIST=YES/NO */

 2: CREATE:  PROCEDURE (PARM) OPTIONS (MAIN);

 3: /* INSTRUCTION */
 4: DECLARE
 5:    INSTR CHAR(80) EXT,
 6:    LIST CHAR(1) DEF INSTR POS(5),
 7:    #_HDGS PIC'Z' DEF INSTR POS(72);

 8: /* PRINT ROUTINE */
 9: DECLARE
10:    PARM CHAR(100),
11:   (HEADING(9),PRINTER) CHAR(132) EXT,
12:    PRINTX ENTRY (PIC'Z');

13: /* FILES */
14: DECLARE
15:    RECORD CHAR(56) BASED (PTR),
16:    BACKDD CHAR(8) STATIC INIT ('SAVESUF'),
17:    PERMDD CHAR(8) STATIC INIT ('SUFFICY'),
18:    PERM FILE RECORD KEYED ENV (INDEXED),
19:    BACKUP FILE RECORD;

20: /* OTHER VARIABLES */
21: DECLARE
22:    CNTR BIN FIXED (31),
23:    PCNTR PIC'ZZZZZ9';


24: /***** INITIALIZATION *****/

25:    CALL INIT (PARM);

26:    /* SET UP HEADINGS */
27:    #_HDGS = 2;
28:    HEADING(1) = PERMDD || 'FILE CREATION ROUTINE';

29:    /* INIT FILES */
30:    OPEN FILE (BACKUP) INPUT TITLE (BACKDD);
31:    OPEN FILE (PERM) OUTPUT TITLE (PERMDD);
32:    ON ENDFILE (BACKUP) GOTO DONE;

33:    /* PRINT DATE */
34:    READ FILE (BACKUP) SET (PTR);
35:    PRINTER = '   DATE OF BACKUP FILE IS ' || RECORD;
36:    CALL PRINTX (1);
37:    PRINTER = ' ';
38:    CALL PRINTX (1);


39: /***** MAIN EXECUTION LOOP *****/

40:    DO CNTR=1 TO 999999;
41:       READ FILE (BACKUP) SET (PTR);
42:       WRITE FILE (PERM) FROM (RECORD) KEYFROM (SUBSTR(RECORD,2));
43:       IF LIST='Y' THEN DO;
44:          PRINTER = '     ' || RECORD;
```

```
/* :CREATE,FILE=SUFFICY,LIST=YES/NO */

 45:           CALL PRINTX (1);
 46:           END;
 47:        END;

 48: DONE:
 49:    PCNTR = CNTR - 1;
 50:    PRINTER = 'NUMBER OF RECORDS IN FILE:  ' || PCNTR;
 51:    CALL PRINTX (3);
 52:    CLOSE FILE (PERM);
 53:    CLOSE FILE (BACKUP);
 54:    CALL EXIT (PARM);

 55: END CREATE;
```

UPDATE -- UPDATE is comprised of three separate programs, one for each of the functions DELETE, INSERT, REWRITE. The names of the routines are "PDS" followed by the first letter of the function ("PDSD" for FUNCTION=DELETE).

FUNCTION=INSERT:

Member Name	PDSI	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- PDSI messages
	SUFFICY	-- Sufficiency file
	any name	-- Sufficiency data cards
Instruction	1 - 4	"PDSI"
	24 - 31	Name of input DD statement

Data cards, when inserting records, contain a complete Sufficiency record. The numeric fields on the data cards are edited prior to insertion. If an error is detected in a record, message is printed for the user and that particular record is not inserted into the Sufficiency file. The data card formats may be found in the publication Highway Information System Volume 1: User Information:

The PDSI program listing follows:

```
/* :UPDATE,FILE=SUFFICY,FUNCTION=INSERT,DDNAME=XXXXX */

 1: /* :UPDATE,FILE=SUFFICY,FUNCTION=INSERT,DDNAME=XXXXX */

 2: PDSI: PROCEDURE(PARM) OPTIONS(MAIN);

 3: /* INSTRUCTION AND PRINT ROUNTINE */
 4:    DECLARE
 5:    DDNAME CHAR(8) DEF INSTR POS(24),
 6:    HEADING(9)                          CHAR(132) EXT,
 7:    #_HDGS                              PIC'Z' DEF INSTR POS(72),
 8:    INSTR                               CHAR(100) EXT,
 9:    PARM                                CHAR(100),
10:    PRINTER                             CHAR(132) EXT,
11:    PRINTX                              ENTRY (PIC'Z');

12: /* DATA INPUT */
13:    DECLARE
14:    ERRORS                              CHAR(1),
15:    STRING_CARD                         CHAR(80),
16:    1 CARD DEF STRING_CARD,
17:         2 KEY                          CHAR(13),
18:         2 DESCR                        CHAR(18),
19:         2 DESIGN_SPEED                 CHAR(2),
20:         2 TERRAIN                      CHAR(1),
21:         2 AVG_SPEED                    CHAR(2),
22:         2 SIGHT_DIST                   CHAR(2),
23:         2 CURVES                       CHAR(2),
24:         2 BRIDGES                      CHAR(1),
25:         2 FOUNDATION                   CHAR(2),
26:         2 SURFACE                      CHAR(2),
27:         2 DRAINAGE                     CHAR(2),
28:         2 SECTION_LENGTH               CHAR(6),
29:         2 MONTH                        CHAR(2),
30:         2 DAY                          CHAR(2),
31:         2 YEAR                         CHAR(2),
32:    DATA FILE RECORD SEQL INPUT;
33:    DECLARE
34:    STRING_SUF                          CHAR(64),
35:    1 SUF DEF STRING_SUF,
36:         2 DUMMY                        CHAR(1),
37:         2 KEY                          CHAR(13),
38:         2 DESCR                        CHAR(18),
39:         2 DESIGN_SPEED                 PIC'ZZ',
40:         2 TERRAIN                      PIC'Z',
41:         2 AVG_SPEED                    PIC'ZZ',
42:         2 SIGHT_DIST                   PIC'ZZ',
43:         2 CURVES                       PIC'ZZ',
44:         2 BRIDGES                      PIC'Z',
45:         2 FOUNDATION                   PIC'ZZ',
46:         2 SURFACE                      PIC'ZZ',
47:         2 DRAINAGE                     PIC'ZZ',
48:         2 SECTION_LENGTH               PIC'ZZZVZZZ',
49:         2 MONTH                        PIC'ZZ',
50:         2 DAY                          PIC'ZZ',
51:         2 YEAR                         PIC'ZZ',
52:         2 DUMMY2                       CHAR(1),
53:    SUFFICY FILE RECORD KEYED ENV(INDEXED);
```

```
/* :UPDATE,FILE=SUFFICY,FUNCTION=INSERT,DDNAME=XXXXX */


 54: /* **INITIALIZATION** */

 55:     CALL INIT(PARM);
 56:     #_HDGS = 2;
 57:     HEADING(1) = '     SUFFICIENCY UPDATE -- FUNCTION INSERT ';
 58:     OPEN
 59:          FILE(DATA) TITLE(DDNAME),
 60:          FILE(SUFFICY) UPDATE DIRECT;
 61:     ON KEY(SUFFICY) BEGIN;
 62:          PRINTER='*** DUPLICATE KEY FOR ATTEMPTED INSERT AT '||CARD.KEY;
 63:          CALL PRINTX(1);
 64:          GO TO READ_DATA;
 65:          END;
 66:     ON ENDFILE(DATA) GO TO CLOSE;


 67: /*   ** EXECUTION LOOP ** */

 68: READ_DATA:
 69:     READ FILE(DATA) INTO(STRING_CARD);
 70:     PRINTER = (5)' '||STRING_CARD;
 71:     CALL PRINTX(2);
 72:     ERRORS = ' ';
 73:     STRING_SUF = ' '||STRING_CARD||' ';

 74: CHECK1: IF CARD.DESIGN_SPEED ¬= '  ' & CARD.AVG_SPEED ¬= '  ' THEN DO;
 75:     IF  SUF.DESIGN_SPEED = 1 | SUF.DESIGN_SPEED = 0 THEN GO TO CHECK2;
 76:     IF (SUF.DESIGN_SPEED = 70 & SUF.AVG_SPEED >= 55) |
 77:        (SUF.DESIGN_SPEED = 60 & SUF.AVG_SPEED >= 50) |
 78:        (SUF.DESIGN_SPEED = 50 & SUF.AVG_SPEED >= 45) |
 79:        (SUF.DESIGN_SPEED = 40 & SUF.AVG_SPEED >= 40)
 80:          THEN GO TO CHECK2;
 81:     PRINTER = '***ERROR - DESIGN SPEED OR AVERAGE SPEED '||
 82:          ' - CHECK PAGE 284 OF 1965 HIGHWAY CAPACITY MANUAL ';
 83:     CALL PRINTX(1);
 84:     ERRORS = 'X';
 85:     END;

 86: CHECK2: IF CARD.TERRAIN = ' ' THEN GO TO CHECK5;
 87:     IF (SUF.TERRAIN < 4 & SUF.TERRAIN > 0 ) THEN GO TO CHECK5;
 88:     PRINTER = '***ERROR - TERRAIN CLASSIFICATION MUST BE 1,2, OR 3';
 89:     CALL PRINTX(1);
 90:     ERRORS = 'X';

 91: CHECK5: IF CARD.SIGHT_DIST = '  ' THEN GO TO CHECK6;
 92:     IF (SUF.SIGHT_DIST < 100 & SUF.SIGHT_DIST >= 0 ) THEN GO TO CHECK6;
 93:     PRINTER = '***ERROR - SIGHT DISTANCE MUST BE 0-99 PERCENT '
 94:     || ' < 1500 FEET';
 95:     CALL PRINTX(1);
 96:     ERRORS = 'X';

 97: CHECK6: IF CARD.CURVES = '  ' THEN GO TO CHECK7;
 98:     IF SUF.CURVES >= 0 & SUF.CURVES < 99 THEN GO TO CHECK7;
 99:     PRINTER = '***ERROR - NUMBER OF UNDER DESIGNED CURVES MUST BE 0-99';
100:     CALL PRINTX(1);
101:     ERRORS = 'X';
```

```
/* :UPDATE,FILE=SUFFICY,FUNCTION=INSERT,DDNAME=XXXXX */

102: CHECK7: IF CARD.BRIDGES = ' ' THEN GO TO CHECK8;
103:     IF SUF.BRIDGES >= 0 & SUF.BRIDGES < 10 THEN GO TO CHECK8;
104:     PRINTER = '***ERROR - NUMBER OF NARROW BRIDGES MUST BE 0-9';
105:     CALL PRINTX(1);
106:     ERRORS = 'X';

107: CHECK8: IF CARD.FOUNDATION = '  ' THEN GO TO CHECK9;
108:     IF SUF.FOUNDATION < 11 & SUF.FOUNDATION >= 0 THEN GO TO CHECK9;
109:     PRINTER = '***ERROR - FOUNDATION RATING MUST BE 0-10';
110:     CALL PRINTX(1);
111:     ERRORS = 'X';

112: CHECK9: IF CARD.SURFACE = '  ' THEN GO TO CHECK10;
113:     IF SUF.SURFACE < 31 & SUF.SURFACE >= 0 THEN GO TO CHECK10;
114:     PRINTER = '***ERROR - SURFACE RATING MUST BE 0-30';
115:     CALL PRINTX(1);
116:     ERRORS = 'X';

117: CHECK10: IF CARD.DRAINAGE = '  ' THEN GO TO CHECK11;
118:     IF SUF.DRAINAGE < 11 & SUF.DRAINAGE >= 0 THEN GO TO CHECK11;
119:     PRINTER = '***ERROR - DRAINAGE RATING MUST BE 0-10';
120:     CALL PRINTX(1);
121:     ERRORS = 'X';

122: CHECK11: IF CARD.SECTION_LENGTH = '      ' THEN GO TO CHECK12;
123:     IF SUF.DESIGN_SPEED ¬= 0 | SUF.DESIGN_SPEED ¬= 1 THEN DO;
124:          PRINTER ='***ERROR - SECTION LENGTH MUST ONLY BE CODED FOR '||
125:          'NON EXISTENT AND UNDER CONSTRUCTION ROADWAY ';
126:          CALL PRINTX(1);
127:          ERRORS = 'X';
128:          END;
129:     IF SUF.SECTION_LENGTH >= 0 & SUF.SECTION_LENGTH < 1000000 THEN
130:     GO TO CHECK12;
131:     PRINTER = '***ERROR - INVALID SECTION LENGTH CODED ';
132:     CALL PRINTX(1);
133:     ERRORS = 'X';

134: CHECK12: IF CARD.MONTH='  ' &CARD.DAY='  '&CARD.YEAR='  '
135:     THEN GO TO ERROR;
136:     IF SUF.MONTH < 1 | SUF.MONTH> 12 THEN DO;
137:          PRINTER = '***ERROR - INVALID MONTH CODED';
138:          CALL PRINTX(1);
139:          ERRORS = 'X';
140:          END;
141:     IF SUF.DAY < 1 | SUF.DAY > 31 THEN DO;
142:          PRINTER = '***ERROR - INVALID DAY OF YEAR CODED';
143:          CALL PRINTX(1);
144:          ERRORS = 'X';
145:          END;
146:     IF SUF.YEAR < 1 | SUF.YEAR> 99 THEN DO;
147:          PRINTER = '***ERROR - INVALID YEAR CODED';
148:          CALL PRINTX(1);
149:          ERRORS = 'X';
150:          END;

151: ERROR:
152:     IF ERRORS = 'X' THEN DO;
```

```
/* :UPDATE,FILE=SUFFICY,FUNCTION=INSERT,DDNAME=XXXXX */

153:     PRINTER = ' PLEASE RESUBMIT THIS REWRITE CARD AFTER CORRECTION '||
154:     'HAS BEEN MADE';
155:     CALL PRINTX(1);
156:     GO TO READ_DATA;
157:     END;

158: INSERT:
159:     WRITE FILE(SUFFICY) FROM(STRING_SUF) KEYFROM(SUF.KEY);
160:     PRINTER = ' '||STRING_SUF;
161:     CALL PRINTX(1);
162:     GO TO READ_DATA;

163: CLOSE: CLOSE
164:     FILE(DATA),
165:     FILE(SUFFICY);
166:     CALL EXIT(PARM);

167: END PDSI;
```

FUNCTION=REWRITE:

Member Name	PDSR
Language	PL/I
Subroutines	PRINTX1
Files	SYSPRINT -- IBM messages PRINTER -- PDSR messages SUFFICY -- Sufficiency file any name -- Sufficiency data cards
Instruction	1 - 4 "PDSR" 24 - 31 Name of input DD statement

Data cards for rewriting, though in the same format as those for inserting, require only those fields being altered to be coded. The numeric fields on the data cards to be inserted are edited. If an error is detected in a record, a message is printed for the user and that particular record is not rewritten into the file. The data card formats may be found in the publication Highway Information System Volume 1: User Information. PDSR also allows the user to rewrite the key field of the record. However, the rewrite data card for changing the key has a particular format consisting of the existing key, an equal sign, and the new key (e.g., P001567+5.123=P001567+4.321). No other data fields may be coded on the "new key" data card.

The PDSR program listing follows:

```
/* : UPDATE,FILE=SUFFICY,FUNCTION=REWRITE,DDNAME=XXXXX */

 1: /* : UPDATE,FILE=SUFFICY,FUNCTION=REWRITE,DDNAME=XXXXX */

 2: PDSR: PROCEDURE(PARM) OPTIONS(MAIN);

 3: /* INSTRUCTION AND PRINT ROUNTINE */
 4:    DECLARE
 5:    DDNAME CHAR(8) DEF INSTR POS(24),
 6:    HEADING(9)                          CHAR(132) EXT,
 7:    #_HDGS                              PIC'Z' DEF INSTR POS(72),
 8:    INSTR                               CHAR(100) EXT,
 9:    PARM                                CHAR(100),
10:    PRINTER                             CHAR(132) EXT,
11:    PRINTX                              ENTRY (PIC'Z');

12: /* DATA INPUT */
13:    DECLARE
14:    ERRORS                              CHAR(1),
15:    STRING_CARD                         CHAR(80),
16:    1 CARD DEF STRING_CARD,
17:         2 KEY                          CHAR(13),
18:         2 DESCR                        CHAR(18),
19:         2 DESIGN_SPEED                 CHAR(2),
20:         2 TERRAIN                      CHAR(1),
21:         2 AVG_SPEED                    CHAR(2),
22:         2 SIGHT_DIST                   CHAR(2),
23:         2 CURVES                       CHAR(2),
24:         2 BRIDGES                      CHAR(1),
25:         2 FOUNDATION                   CHAR(2),
26:         2 SURFACE                      CHAR(2),
27:         2 DRAINAGE                     CHAR(2),
28:         2 SECTION_LENGTH               CHAR(6),
29:         2 MONTH                        CHAR(2),
30:         2 DAY                          CHAR(2),
31:         2 YEAR                         CHAR(2),
32:    C(62)                               CHAR(1) DEF STRING_CARD,
33:    DATA FILE RECORD SEQL INPUT;

34: /* SUFFICIENCY FILE */
35:    DECLARE
36:    S(63)                               CHAR(1) DEF STRING_SUF POS(2),
37:    STRING_SUF                          CHAR(64),
38:    1 SUF DEF STRING_SUF,
39:         2 DUMMY                        CHAR(1),
40:         2 KEY                          CHAR(13),
41:         2 DESCR                        CHAR(18),
42:         2 DESIGN_SPEED                 PIC'ZZ',
43:         2 TERRAIN                      PIC'Z',
44:         2 AVG_SPEED                    PIC'ZZ',
45:         2 SIGHT_DIST                   PIC'ZZ',
46:         2 CURVES                       PIC'ZZ',
47:         2 BRIDGES                      PIC'Z',
48:         2 FOUNDATION                   PIC'ZZ',
49:         2 SURFACE                      PIC'ZZ',
50:         2 DRAINAGE                     PIC'ZZ',
51:         2 SECTION_LENGTH               PIC'ZZZVZZZ',
52:         2 MONTH                        PIC'ZZ',
53:         2 DAY                          PIC'ZZ',
54:         2 YEAR                         PIC'ZZ',
```

```
/* : UPDATE,FILE=SUFFICY,FUNCTION=REWRITE,DDNAME=XXXXX */

 55:         2 DUMMY2                          CHAR(1),
 56:     SUFFICY FILE RECORD KEYED ENV(INDEXED);


 57: /* **INITIALIZATION** */

 58:     CALL INIT(PARM);
 59:     #_HDGS = 2;
 60:     HEADING(1) = '    SUFFICIENCY UPDATE -- FUNCTION REWRITE ';
 61:     OPEN
 62:          FILE(DATA) TITLE(DDNAME),
 63:          FILE(SUFFICY) UPDATE DIRECT;
 64:     ON KEY(SUFFICY) BEGIN;
 65:          PRINTER = '*** NO RECORD FOR ATTEMPTED REWRITE AT '||CARD.KEY;
 66:          CALL PRINTX(1);
 67:          GO TO READ_DATA;
 68:          END;
 69:     ON ENDFILE(DATA) GO TO CLOSE;


 70: /*   ** EXECUTION LOOP ** */

 71: READ_DATA:
 72:     READ FILE(DATA) INTO(STRING_CARD);
 73:     PRINTER = (5)' '||STRING_CARD;
 74:     CALL PRINTX(2);
 75:     READ FILE(SUFFICY) INTO(STRING_SUF) KEY(CARD.KEY);
 76:     ERRORS = ' ';
 77:
 78:     DO I = 13 TO 62;
 79:          IF C(I) ¬= ' ' THEN DO;
 80:               IF C(I) = '$' THEN C(I) = ' ';
 81:               S(I) = C(I);
 82:               END;
 83:          END;

 84: CHECK1: IF CARD.DESIGN_SPEED ¬= '  ' & CARD.AVG_SPEED ¬= '  ' THEN DO;
 85:     IF  SUF.DESIGN_SPEED = 1 | SUF.DESIGN_SPEED = 0 THEN GO TO CHECK2;
 86:     IF CARD.DESIGN_SPEED = '$$'& CARD.AVG_SPEED ='$$' THEN GO TO CHECK4;
 87:     IF (SUF.DESIGN_SPEED = 70 & SUF.AVG_SPEED >= 55) |
 88:        (SUF.DESIGN_SPEED = 60 & SUF.AVG_SPEED >= 50) |
 89:        (SUF.DESIGN_SPEED = 50 & SUF.AVG_SPEED >= 45) |
 90:        (SUF.DESIGN_SPEED = 40 & SUF.AVG_SPEED >= 40)
 91:       THEN GO TO CHECK4;
 92:     PRINTER = '***ERROR - DESIGN SPEED AR AVERAGE SPEED '||
 93:          ' - CHECK PAGE 284 OF 1965 HIGHWAY CAPACITY MANUAL ';
 94:     CALL PRINTX(1);
 95:     ERRORS = 'X';
 96:     END;

 97: CHECK2: IF CARD.DESIGN_SPEED = '  ' | CARD.DESIGN_SPEED = '$$' |
 98:     SUF.DESIGN_SPEED = 1 | SUF.DESIGN_SPEED = 0 THEN GO TO CHECK3;
 99:     IF (SUF.DESIGN_SPEED = 70 & SUF.AVG_SPEED >= 55) |
100:        (SUF.DESIGN_SPEED = 60 & SUF.AVG_SPEED >= 50) |
101:        (SUF.DESIGN_SPEED = 50 & SUF.AVG_SPEED >= 45) |
102:        (SUF.DESIGN_SPEED = 40 & SUF.AVG_SPEED >= 40)
103:          THEN GO TO CHECK3;
104:     PRINTER = '***ERROR - DESIGN SPEED AR AVERAGE SPEED '||
```

```
/* : UPDATE,FILE=SUFFICY,FUNCTION=REWRITE,DDNAME=XXXXX */

105:          ' - CHECK PAGE 284 OF 1965 HIGHWAY CAPACITY MANUAL ';
106:     CALL PRINTX(1);
107:     ERRORS = 'X';

108: CHECK3: IF CARD.AVG_SPEED = '   ' | CARD.AVG_SPEED ='$$'
109:     THEN GO TO CHECK4;
110:     IF ( SUF.AVG_SPEED >= 55 & SUF.DESIGN_SPEED = 70) |
111:          ( SUF.AVG_SPEED = 50 & SUF.DESIGN_SPEED = 60) |
112:          ( SUF.AVG_SPEED = 45 & SUF.DESIGN_SPEED = 50) |
113:          ( SUF.AVG_SPEED >= 40 & SUF.DESIGN_SPEED = 40)
114:     THEN GO TO CHECK4;
115:     PRINTER = '***ERROR - DESIGN SPEED AR AVERAGE SPEED '||
116:          ' - CHECK PAGE 284 OF 1965 HIGHWAY CAPACITY MANUAL ';
117:     CALL PRINTX(1);
118:     ERRORS = 'X';

119: CHECK4: IF CARD.TERRAIN = ' ' | CARD.TERRAIN = '$' THEN GO TO CHECK5;
120:     IF (SUF.TERRAIN < 4 & SUF.TERRAIN > 0 ) THEN GO TO CHECK5;
121:     PRINTER = '***ERROR - TERRAIN CLASSIFICATION MUST BE 1,2, OR 3';
122:     CALL PRINTX(1);
123:     ERRORS = 'X';

124: CHECK5: IF CARD.SIGHT_DIST = '   ' | CARD.SIGHT_DIST = '$$'
125:     THEN GO TO CHECK6;
126:     IF (SUF.SIGHT_DIST < 100 & SUF.SIGHT_DIST >= 0 ) THEN GO TO CHECK6;
127:     PRINTER = '***ERROR - SIGHT DISTANCE MUST BE 0-99 PERCENT '
128:     || ' < 1500 FEET';
129:     CALL PRINTX(1);
130:     ERRORS = 'X';

131: CHECK6: IF CARD.CURVES = '   ' | CARD.CURVES = '$$' THEN GO TO CHECK7;
132:     IF SUF.CURVES >= 0 & SUF.CURVES < 99 THEN GO TO CHECK7;
133:     PRINTER = '***ERROR - NUMBER OF UNDER DESIGNED CURVES MUST BE 0-99';
134:     CALL PRINTX(1);
135:     ERRORS = 'X';

136: CHECK7: IF CARD.BRIDGES = ' ' | CARD.BRIDGES = '$' THEN GO TO CHECK8;
137:     IF SUF.BRIDGES >= 0 & SUF.BRIDGES < 10 THEN GO TO CHECK8;
138:     PRINTER = '***ERROR - NUMBER OF NARROW BRIDGES MUST BE 0-9';
139:     CALL PRINTX(1);
140:     ERRORS = 'X';

141: CHECK8: IF CARD.FOUNDATION = '   ' | CARD.FOUNDATION='$$'
142:     THEN GO TO CHECK9;
143:     IF SUF.FOUNDATION < 11 & SUF.FOUNDATION >= 0 THEN GO TO CHECK9;
144:     PRINTER = '***ERROR - FOUNDATION RATING MUST BE 0-10';
145:     CALL PRINTX(1);
146:     ERRORS = 'X';

147: CHECK9: IF CARD.SURFACE='   ' | CARD.SURFACE='$$' THEN GO TO CHECK10;
148:     IF SUF.SURFACE < 31 & SUF.SURFACE >= 0 THEN GO TO CHECK10;
149:     PRINTER = '***ERROR - SURFACE RATING MUST BE 0-30';
150:     CALL PRINTX(1);
151:     ERRORS = 'X';

152: CHECK10: IF CARD.DRAINAGE='   ' | CARD.DRAINAGE='$$' THEN GO TO CHECK11;
153:     IF SUF.DRAINAGE < 11 & SUF.DRAINAGE >= 0 THEN GO TO CHECK11;
154:     PRINTER = '***ERROR - DRAINAGE RATING MUST BE 0-10';
```

```
/* : UPDATE,FILE=SUFFICY,FUNCTION=REWRITE,DDNAME=XXXXX */

155:    CALL PRINTX(1);
156:    ERRORS = 'X';

157: CHECK11: IF CARD.SECTION_LENGTH = '      ' | CARD.SECTION_LENGTH =
158:    '$$$$$$' THEN GO TO CHECK12;
159:    IF SUF.DESIGN_SPEED ¬= 0 | SUF.DESIGN_SPEED ¬= 1 THEN DO;
160:         PRINTER ='***ERROR - SECTION LENGTH MUST ONLY BE CODED FOR '||
161:         'NON EXISTENT AND UNDER CONSTRUCTION ROADWAY ';
162:         CALL PRINTX(1);
163:         ERRORS = 'X';
164:         END;
165:    IF SUF.SECTION_LENGTH >= 0 & SUF.SECTION_LENGTH < 1000000 THEN
166:    GO TO CHECK12;
167:    PRINTER = '***ERROR - INVALID SECTION LENGTH CODED ';
168:    CALL PRINTX(1);
169:    ERRORS = 'X';

170: CHECK12: IF CARD.MONTH='  ' &CARD.DAY='  'CARD.YEAR='  '
171:    THEN GO TO CHANGE_KEY;
172:    IF CARD.MONTH='$$' | CARD.DAY='$$' | CARD.YEAR='$$'
173:    THEN GO TO CHANGE_KEY;
174:    IF SUF.MONTH < 1 | SUF.MONTH> 12 THEN DO;
175:         PRINTER = '***ERROR - INVALID MONTH CODED';
176:         CALL PRINTX(1);
177:         ERRORS = 'X';
178:         END;
179:    IF SUF.DAY < 1 | SUF.DAY > 31 THEN DO;
180:         PRINTER = '***ERROR - INVALID DAY OF YEAR CODED';
181:         CALL PRINTX(1);
182:         ERRORS = 'X';
183:         END;
184:    IF SUF.YEAR < 1 | SUF.YEAR> 99 THEN DO;
185:         PRINTER = '***ERROR - INVALID YEAR CODED';
186:         CALL PRINTX(1);
187:         ERRORS = 'X';
188:         END;

189: CHANGE_KEY:
190:    IF SUBSTR(CARD.DESCR,1,1) = '=' THEN DO;
191:         DELETE FILE(SUFFICY) KEY(SUF.KEY);
192:         SUF.KEY = SUBSTR(CARD.DESCR,2,13);
193:         WRITE FILE(SUFFICY) FROM(STRING_SUF) KEYFROM(SUF.KEY);
194:         GO TO READ_DATA;
195:         END;

196: ERROR:
197:    IF ERRORS = 'X' THEN DO;
198:    PRINTER = ' PLEASE RESUBMIT THIS REWRITE CARD AFTER CORRECTION '||
199:    'HAS BEEN MADE';
200:    CALL PRINTX(1);
201:    GO TO READ_DATA;
202:    END;

203: REWRITE:
204:    REWRITE FILE(SUFFICY) FROM(STRING_SUF) KEY(SUF.KEY);
205:    PRINTER = ' '||STRING_SUF;
206:    CALL PRINTX(1);
207:    GO TO READ_DATA;
```

```
/* : UPDATE,FILE=SUFFICY,FUNCTION=REWRITE,DDNAME=XXXXX */

208: CLOSE: CLOSE
209:    FILE(DATA),
210:    FILE(SUFFICY);
211:    CALL EXIT(PARM);

212: END PDSR;
```

FUNCTION=DELETE:

Member Name	PDSD	
Language	PL/I	
Subroutines	PRINTX1	
Files	SYSPRINT	-- IBM messages
	PRINTER	-- PDSD messages
	SUFFICY	-- Sufficiency file
	any name	-- Sufficiency data cards
Instruction	1 - 4	"PDSD"
	24 - 31	Name of input DD statement

This program operates with a direct update Sufficiency file. The update data cards containing the record keys in columns 1-13 are read by the program. An attempt is then made to delete those records whose keys were coded on the update cards. If a record does not exist in the Sufficiency file corresponding to a data card, an error message is printed. Each data card is printed as it is read.

The PDSD program listing follows:

```
/* :UPDATE,FILE=SUFFICY,FUNCTION=DELETE,DDNAME=XXXXX */

 1: /* :UPDATE,FILE=SUFFICY,FUNCTION=DELETE,DDNAME=XXXXX */

 2: DELETE:  PROCEDURE (PARM) OPTIONS (MAIN);

 3: /* INSTRUCTION */
 4: DECLARE
 5:    INSTR CHAR(80) EXT,
 6:    #_HDGS PIC'Z' DEF INSTR POS(72),
 7:    DDNAME CHAR(8) DEF INSTR POS(24);

 8: /* PRINT ROUTINE */
 9: DECLARE
10:    PARM CHAR(100),
11:   (HEADING(9),PRINTER) CHAR(132) EXT,
12:    PRINTX ENTRY (PIC'Z');

13: /* PERMANENT FILE */
14: DECLARE
15:    PERMDD CHAR(8) STATIC INIT ('SUFFICY'),
16:    PERM FILE RECORD KEYED ENV (INDEXED);
17: DECLARE
18:    DATA FILE RECORD,
19:    KEY CHAR(13) BASED (PTR_DATA);


20: /***** INITIALIZATION *****/
21:    CALL INIT (PARM);

22:    /* SET UP HEADINGS */
23:    #_HDGS = 2;
24:    HEADING(1) = PERMDD || 'FILE UPDATE -- DELETION OF RECORDS';

25:    /* OPEN FILES */
26:    ON UNDEFINEDFILE (DATA) BEGIN;
27:       PRINTER = '*** ' || DDNAME || ' DD STATEMENT MISSING';
28:       CALL PRINTX (3);
29:       GOTO RETURN;
30:       END;
31:    OPEN FILE (DATA) INPUT RECORD TITLE (DDNAME);
32:    ON ENDFILE (DATA) GOTO CLOSE;
33:    OPEN FILE (PERM) UPDATE DIRECT TITLE (PERMDD);
34:    ON KEY (PERM) BEGIN;
35:       PRINTER = '*** RECORD DOES NOT EXIST IN FILE';
36:       CALL PRINTX (1);
37:       GOTO READ_DATA;
38:       END;


39: /***** MAIN EXECUTION LOOP *****/

40: READ_DATA:
41:    READ FILE (DATA) SET (PTR_DATA);
42:    PRINTER = '     ' || KEY;
43:    CALL PRINTX (2);
44:    DELETE FILE (PERM) KEY (KEY);
45:    GOTO READ_DATA;

46: CLOSE:
```

```
/* :UPDATE,FILE=SUFFICY,FUNCTION=DELETE,DDNAME=XXXXX */

 47:     CLOSE FILE (PERM);
 48:     CLOSE FILE (DATA);
 49:     CALL EXIT (PARM);

 50: RETURN:

 51: END DELETE;
```

CHAPTER 2-VI

PRELIMINARY STUDY OF RETRIEVAL OF ROADWAY GEOMETRIC INFORMATION

Introduction

This chapter presents a description of the files and programs comprising the preliminary study of roadway geometrics. It is designed for utilization with the publication Highway Information System Volume 1: User Information.

Geometric Vertical File Description

Organization . . . . . Sequential
Record Length 80

The format of a Geometric Vertical record is shown in Table 2-VI-I.

Geometric Horizontal File Description

Organization . . . . . Sequential
Record Length 80

The format of a Geometric Horizontal record is shown in Table 2-VI-II.

Program Descriptions

The programs written for the preliminary geometrics study were written in FORTRAN for use on Montana State University's XDS Sigma 7 computer. A description of each of the programs follows.

Vertical Program -- The vertical program reads, from a data card, a starting and ending location (stationing) between which the user wants to know the relative difference in elevation. The program reads the data file until it gets to the route that is also specified by the user on the same data card.

TABLE 2-VI-I

GEOMETRIC VERTICAL RECORDS

Columns	Contents	Remarks
1	Route System	See Note 1 below.
2-5	Route Number	See Note 2 below.
6-15	Stationing	See Note 3 below.
16	Type Code	See Note 4 below.
17-20	Blank	
----------	Intersection Records ----------	
21	Route System on intersecting route	See Note 1 below.
22-25	Route Number on intersecting route	See Note 2 below.
26-35	Stationing on intersecting route	See Note 3 below.
36-80	Blank	
----------	Equation Records ----------	
21-25	Blank	
26-35	Ahead Stationing	See Note 3 below.
36-80	Blank	
----------	Vertical Curve Record ----------	
21-80	Blank	
----------	Vertical Tangent Record ----------	
21-25	Blank	
26-37	Grade	See Note 5 below.
38-80	Blank	
----------	Equation-in-Grade Record ----------	
21-25	Blank	
26-37	Difference in Elevation	See Note 5 below.
38-80	Blank	

Notes:

1. Route System is "P" for Primary and "S" for Secondary.
2. Route Number is right-justified in 4-digit field.
3. Stationing is right-justified in a 10-digit field, with a decimal point in column 8 of the field.
4. Type code is "I" for intersection records, "E" for equation records, "C" for vertical curve records, "T" for vertical tangent records and "D" for equation-in-grade records.
5. The grade and difference in elevation are right-justified in a 12-digit field. A decimal point appears in column 8 of the field.

TABLE 2-VI-II

GEOMETRIC HORIZONTAL RECORDS

Columns	Contents	Remarks
1	Route System	See note 1 below.
2-5	Route Number	See note 2 below.
6-15	Stationing	See note 3 below.
16	Type Code	See note 4 below.
17-20	Blank	
----------	Intersection Records ----------	
21	Route System on intersecting route	See note 1 below.
22-25	Route Number on intersecting route	See note 2 below.
26-35	Blank	
36-45	Stationing on intersecting route	See note 3 below.
46-80	Blank	
----------	Equation Records ----------	
21-35	Blank	
36-45	Ahead stationing	See note 3 below.
46-80	Blank	
----------	Horizontal Tangent Records ----------	
21	Primary Direction	See note 5 below.
22-25	Degrees	See note 6 below.
26-30	Minutes	See note 6 below.
31-35	Seconds	See note 6 below.
36	Secondary Direction	See note 7 below.
37-80	Blank	
----------	Horizontal Curve Records ----------	
21	Blank	
22-25	Degrees	See note 6 below.
26-30	Minutes	See note 6 below.
31-35	Seconds	See note 6 below.
36	Direction	See note 8 below.
37-80	Blank	
----------	Final Route Entry Records ----------	
21-80	Blank	

Notes:
1. Route System is "P" for Primary and "S" for Secondary.
2. Route Number is right-justified in 4-digit field.
3. Stationing is right-justified in a 10-digit field, with a decimal point in column 8 of the field.
4. Type code is "I" for intersection records, "E" for equation records, "C" for horizontal curve records, "T" for horizontal tangents records, and "F" for final route entry records.
5. "N" for north and "S" for south.
6. Right-justified in field.
7. "E" for east and "W" for west.
8. "L" for left and "R" for right.

The program then sequentially reads the data file until it finds a record whose stationing is greater than or equal to the starting location. If the record stationing is greater, it calculates the difference in elevation and the horizontal distance between the start location and the record location and sets the new location equal to the record's stationing. If the record stationing is equal to the start location, the new location is set equal to the start location.

A new record is read and checked to see if it is beyond the end location. If not, the difference in elevation is found between the previous record read (called the new location) and the new record and added to the total difference in elevation. The new location is set to the new record the next record is read and the process is repeated.

The repetitive process ceases whenever the new record location is beyond the end location. The difference in elevation between the new location and the end location is then calculated and added to the total elevation. The horizontal distance traveled and the relative difference in elevation between the start location and the end location is then printed out. Then the program checks for more "command" data cards and when they are exhausted, a final printout is made telling the total horizontal distance traversed and the total difference in elevation covering all of the "command" data cards.

Subroutines are used to do the calculations of the difference in elevation between the new location and the new record. The vertical tangent subroutine receives the location of the beginning of the tangent and a final location. The subroutine returns the calculated relative distance and the relative difference in elevation between the two locations. The vertical curve subroutine receives the location of the beginning of the curve and a final location and returns the same information as the tangent subroutine. All records read are available to the subroutines.

The user must define to the program whether he wants processing to start at the beginning of the route and progress toward the end or start at the end of a route and progress toward the beginning of the route. If the vertical program is furnished with the start location and end location both negative, the program assumes the points in question occur in opposite order to the records in the file.

A program listing of this vertical algorithm follows:

```
      INTEGER SYS,STSYS,STNUM
      INTEGER  TV,CV,SS,TDIM,TYP
      REAL LOC,LEN,NEWLOC
      DIMENSION LOC(4),TYP(4),LEN(4),GRADE(4)
      GLOBAL LOC,TYP,LEN,GRADE,TDIM,II,TV,CV,SS
      DATA II/'I   '/,TV/'T   '/,CV/'C   '/,SS/'E   '/
      DATA ID/'D   '/
      I = 1
      TDIM = 4
      ELEVATION = 0.0
      DISTANCE = 0.0
2     TOTELEV = 0.0
      TOTDIST = 0.0
      REVDIR = 0.0
      READ(105,4,END=966)STSYS,STNUM,STLOC,ENDLOC
4     FORMAT(A1,I3,1X,F10.1,1X,F10.1)
      MERR = 5
      IF( (STLOC.GE.0.0).AND.(ENDLOC.GE.0.0) ) GO TO 5
      TEMP = STLOC
      STLOC = - ENDLOC
      ENDLOC = - TEMP
      REVDIR = 9.9
5     READ(109,6,END=900)SYS,NUM,LOC(I),TYP(I),ISYS,INUM,
     1 GRADE(I)
6     FORMAT(A1,I4,F10.2,A1,4X,A1,I4,F15.2)
      IF( SYS.NE.STSYS     ) GO TO 5
      IF( NUM.NE.STNUM     ) GO TO 5
8     IF( LOC(I).GT.STLOC ) GO TO 5
7     I = I + 1
      IF( I.GT.TDIM ) I = 1
      MERR = 7
10    READ(109,6,END=900)SYS,NUM,LOC(I),TYP(I),ISYS,INUM,
     1 GRADE(I)
      IF( TYP(I).EQ.II ) GO TO 10
      IF( TYP(I).NE.ID ) GO TO 13
      TOTELEV = TOTELEV + GRADE(I)
      GO TO 10
13    IF( LOC(I).GE.STLOC ) GO TO 15
C..       GO TO READ NEXT RECORD
      IF( TYP(I).NE.SS ) GO TO 7
      TEMP = GRADE(I) - LOC(I)
11    IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      LOC(IT) = TEMP + LOC(IT)
      GO TO 10
C..             LOC(I).GE.START LOCATION
15    IF( TYP(I).NE.SS ) GO TO 20
      TEMP = GRADE(I) - LOC(I)
      STLOC = TEMP + STLOC
      GO TO 11
C..             LOC(I).GE.STLOC & LOC(I) NOT SS
```

```
20    IF( LOC(I).EQ.STLOC ) GO TO 30
C..             LOC(I).GT.STLOC
      IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      IF(TYP(IT).EQ.TV)CALL VTAN(IT,STLOC,DIFELEV1,DIFLOC1)
     1 CALL VTAN(IT,LOC(I),DIFELEV2,DIFLOC2)  GO TO 25
      IF(TYP(IT).EQ.CV)CALL VCURVE(IT,STLOC,DIFELEV1,DIFLOC1)
     1 CALL VCURVE(IT,LOC(I),DIFELEV2,DIFLOC2)  GO TO 25
      WRITE(108,23)TYP(I)
23    FORMAT(1HO,'ERROR /230 DIDN''T HAVE VC OR VT AT START',
     1'   THE ENTRY WAS ',A4)
      GO TO 999
25    TOTELEV = DIFELEV2 - DIFELEV1
      NEWLOC = STLOC + DIFLOC2 - DIFLOC1
      TOTDIST = DIFLOC2 - DIFLOC1
      GO TO 100
C..             LOC(I).EQ.STLOC
30    NEWLOC = STLOC
100   MERR = 110
105   I = I + 1
      IF( I.GT.TDIM ) I = 1
110   READ(109,6,END=900)SYS,NUM,LOC(I),TYP(I),ISYS,INUM,
     1 GRADE(I)
      IF( TYP(I).NE.ID ) GO TO 113
      TOTELEV = TOTELEV + GRADE(I)
      GO TO 110
113   IF( TYP(I).NE.SS ) GO TO 120
      TEMP = GRADE(I) - LOC(I)
111   IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      LOC(IT) = TEMP + LOC(IT)
      NEWLOC = TEMP + NEWLOC
      GO TO 110
120   IF( ENDLOC.EQ.LOC(I) ) GO TO 200
      IF( (SYS.NE.STSYS).OR.(NUM.NE.STNUM) ) GO TO 200
      IF( TYP(I).EQ.II ) GO TO 110
      IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      IF(TYP(IT).EQ.TV)CALL VTAN(IT,LOC(I),DIFELEV1,DIFLOC1)
     1   GO TO 125
      IF(TYP(IT).EQ.CV)CALL VCURVE(IT,LOC(I),DIFELEV1,DIFLOC1)
     1   GO TO 125
      WRITE(108,123)TYP(IT)
123   FORMAT(1HO,'ERROR /1230 DIDN''T HAVE VC OR VT ',
     1'   THE ENTRY WAS ',A4)
      GO TO 999
125   NEWLOC = NEWLOC + DIFLOC1
      TOTDIST = TOTDIST + DIFLOC1
      TOTELEV = TOTELEV + DIFELEV1
      GO TO 100
200   IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
```

```
      IF(TYP(IT).EQ.TV)CALL VTAN(IT,ENDLOC,DIFELEV1,DIFLOC1)
     1   GO TO 225
      IF(TYP(IT).EQ.CV)CALL VCURVE(IT,ENDLOC,DIFELEV1,DIFLOC1)
     1   GO TO 225
      WRITE(108,223)TYP(I)
223   FORMAT(1HO,'ERROR /2230 DIDN''T HAVE VC OR VT ',
     1'   THE ENTRY WAS ',A4)
      GO TO 999
225   TOTELEV = TOTELEV + DIFELEV1
      TOTDIST = TOTDIST + DIFLOC1
      NEWLOC = NEWLOC + DIFLOC1
      IF( ABS(NEWLOC-ENDLOC).GT.0.001 ) GO TO 910
      IF( REVDIR.EQ.0.0 ) GO TO 226
      TEMP = STLOC
      STLOC = ENDLOC
      ENDLOC = TEMP
      TOTELEV = - TOTELEV
226   IF( (SYS.NE.STSYS).AND.(NUM.NE.STNUM) ) GO TO 908
      ELEVATION = ELEVATION + TOTELEV
      DISTANCE = DISTANCE + TOTDIST
      WRITE(108,210)STSYS,STNUM,STLOC,ENDLOC,TOTDIST,TOTELEV
210   FORMAT(/,1H ,'ROUTE NUMO ',A1,I3,3X,'START',
     1' LOCO',F8.1,3X,'END LOCO',F8.1,/,1H ,
     2'DISTANCE TRAVELEDO ',F15.4,2X,'FEET',/,1H ,
     3'DIFFERENCE IN ELEVATIONO ',F15.4,2X,'FEET',/)
      REWIND 109
      GO TO 2
900   WRITE(108,901)MERR,SYS,NUM,LOC(I),TYP(I),LEN(I),GRADE(I)
901   FORMAT(1HO,'ERROR /'I3'0 EOF ENCOUNTERED PREMATURELY'
     1,/,1H ,'LAST RECORD READO ',
     2A1,I3,F8.1,A2,F7.1,F6.2)
      GO TO 999
908   WRITE(108,909) MERR,STSYS,SYS,STNUM,NUM
909   FORMAT(1HO,'ERROR /',I3,'0 START SYSTEM',A1,2X,
     1'FINAL SYSTEMO ',A1,/,1H ,
     2'START NUMBERO ',I3,3X,'FINAL NUMBERO ',I3)
      GO TO 999
910   WRITE(108,911) MERR,NEWLOC,ENDLOC
911   FORMAT(1HO,'ERROR /',I3,'0 CALC LOCO ',F10.1,5X,
     1'BEYOND END LOCO ',F10.1)
999   STOP ' ABORTED IN MAINLINE'
966   WRITE(108,967)DISTANCE,ELEVATION
967   FORMAT(//,1H ,'TOTAL DISTANCE TRAVELEDO ',F15.4,
     12X,'FEET',/,1H ,'TOTAL DIFFERENCE IN ',
     2'ELEVATIONO ',F15.4,2X,'FEET')
      STOP 'NORMAL STOP IN MAINLINE'
      END
C..
C..
      SUBROUTINE VTAN(I,ENDLOC,DIFELEV,XSTA)
      INTEGER  TV,CV,SS,TDIM,TYP
      REAL LOC
```

```
      DIMENSION LOC(4),TYP(4) ,GRADE(4)
      GLOBAL LOC,TYP,GRADE,TDIM,II,TV,CV,SS
      XSTA = ENDLOC - LOC(I)
      DIFELEV = XSTA * GRADE(I) / 100.0
990   RETURN
      END
C..
C..
      SUBROUTINE VCURVE(K,ENDLOC,DIFELEV,XSTA)
      INTEGER  TV,CV,SS,TDIM,TYP
      REAL LOC
      DIMENSION LOC(4),TYP(4),GRADE(4)
      GLOBAL LOC,TYP,GRADE,TDIM,II,TV,CV,SS
      IT = K - 1
      IF( K.EQ.1 ) IT = TDIM
      IT1 = K + 1
      IF( K.EQ.TDIM ) IT1 = 1
100   IF( (TYP(IT).NE.TV).AND.(TYP(IT1).NE.TV) )
     1 GO TO 901
110   XSTA = ENDLOC - LOC(K)
111   XLSTA = LOC(IT1) - LOC(K)
121   DIFELEV=((GRADE(IT1)-GRADE(IT))/(200.*XLSTA))
     1*XSTA**2 +(GRADE(IT)*XSTA)/100.0
990   RETURN
901   WRITE(108,902) TYP(IT),TYP(IT1)
902   FORMAT(/,' ERROR VC/1000TYPES NOT EQUAL TO T',2X,2A1)
999   STOP 'ABORTED DUE TO ERROR IN VCURVE SUBROUTINE'
      END
```

Horizontal program -- The program algorithm for the horizontal program is identical to that for the vertical program except that now x and y distances are calculated instead of elevations. Due to the size of the program, the reverse stationing feature present in the vertical program is not implemented. A listing of the horizontal program follows:

```
      INTEGER SYS,STSYS,STNUM
      INTEGER HT,HC,SPIR,SS,TDIM,TYP,DIR,EW,RIGHT
      INTEGER SOUTH,EAST,WEST
      REAL LOC,NEWLOC
      DIMENSION LOC(10),TYP(10),NS(10),DEG(10,3),EW(10)
      DATA II/'I    '/,HT/'T    '/,HC/'C    '/,SPIR/'S    '/
      DATA SS/'E    '/,RIGHT/'R    '/,LEFT/'L    '/
      DATA NORTH/'N    '/,SOUTH/'S    '/,EAST/'E    '/
      DATA WEST/'W    '/
      TDIM = 10
      PI = 3.14159265359
      DISTANCE = 0.0
      DISTX    = 0.0
      DISTY    = 0.0
2     I = 1
      TOTX = 0.0
      TOTY = 0.0
      TOTDIST = 0.0
      AZIMUTH = 0.0
      AZIMADJ = 0.0
      READ(105,3,END=966)STSYS,STNUM,STLOC,ENDLOC
3     FORMAT(A1,I3,1X,F10.1,1X,F10.1)
4     MERR = 5
5     READ(109,6,END=900) SYS,NUM,LOC(I),TYP(I),
     1 NS(I),(DEG(I,K),K=1,3),EW(I),AHSTA
6     FORMAT(A1,1X,I3,F10.2,A1,4X,A1,I4,I5,I5,A1,F9.2)
      IF( SYS.NE.STSYS    ) GO TO 5
      IF( NUM.NE.STNUM    ) GO TO 5
8     IF( LOC(I).GT.STLOC ) GO TO 5
7     I = I + 1
      IF( I.GT.TDIM ) I = 1
      MERR = 10
10    READ(109,6,END=900) SYS,NUM,LOC(I),TYP(I),
     1 NS(I),(DEG(I,K),K=1,3),EW(I),AHSTA
      IF( TYP(I).EQ.II ) GO TO 10
      IF( LOC(I).GE.STLOC ) GO TO 15
C..       GO TO READ NEXT RECORD
      IF( TYP(I).NE.SS ) GO TO 10
      TEMP = AHSTA - LOC(I)
11    IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      LOC(IT) = TEMP + LOC(IT)
      GO TO 10
C..             LOC(I).GE.START LOCATION
15    IF( TYP(I).NE.SS ) GO TO 20
      TEMP = AHSTA - LOC(I)
      STLOC = TEMP + STLOC
      GO TO 11
C..             LOC(I).GE.STLOC & LOC(I) NOT SS
20    IF( LOC(I).EQ.STLOC ) GO TO 90
C..            LOC(I).GT.STLOC
```

```
      IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      IF( TYP(IT).NE.HT ) GO TO 25
      CALL HTAN(IT,STLOC,DIFLOC1,DIFX1,DIFY1)
      CALL HTAN(IT,LOC(I),DIFLOC2,DIFX2,DIFY2)
      GO TO 70
25    IF( TYP(IT).NE.HC ) GO TO 30
      CALL HCURVE(IT,STLOC,DIFLOC1,DIFX1,DIFY1)
      CALL HCURVE(IT,LOC(I),DIFLOC2,DIFX2,DIFY2)
      GO TO 70
30    IF( TYP(IT).NE.SPIR ) GO TO 35
      CALL SPIRAL(IT,STLOC,DIFLOC1,DIFX1,DIFY1)
      CALL SPIRAL(IT,LOC(I),DIFLOC2,DIFX2,DIFY2)
      GO TO 70
35    WRITE(108,23)TYP(I)
23    FORMAT(1HO,'ERROR /230 DIDN''T HAVE VC OR VT AT START',
     1'   THE ENTRY WAS ',A4)
      GO TO 999
70    TOTX = DIFX2 - DIFX1
      TOTY = DIFY2 - DIFY1
      TOTDIST = DIFLOC2 - DIFLOC1
      NEWLOC = STLOC + DIFLOC2 - DIFLOC1
      WRITE(108,72) SYS,NUM,LOC(I),TYP(I),TOTX,TOTY
72    FORMAT(1H ,'RT NO.0',A1,I3,5X,'STATIONO',F10.2
     1,5X,'START OFO',A1,5X,/,1H ,'X =',F10.2
     2,5X,'Y =',F10.2)
      GO TO 100
C..             LOC(I).EQ.STLOC
90    NEWLOC = STLOC
100   MERR = 110
105   I = I + 1
      IF( I.GT.TDIM ) I = 1
110   READ(109,6,END=900) SYS,NUM,LOC(I),TYP(I),
     1 NS(I),(DEG(I,K),K=1,3),EW(I),AHSTA
      IF( TYP(I).NE.SS ) GO TO 120
      TEMP = AHSTA - LOC(I)
111   IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      LOC(IT) = TEMP + LOC(IT)
      NEWLOC = TEMP + NEWLOC
      GO TO 110
120   IF( ENDLOC.EQ.LOC(I) ) GO TO 200
      IF( (SYS.NE.STSYS).OR.(NUM.NE.STNUM) ) GO TO 200
      IF( TYP(I).EQ.II ) GO TO 110
      IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      IF( TYP(IT).NE.HT ) GO TO 125
      CALL HTAN(IT,LOC(I),DIFLOC1,DIFX1,DIFY1)
      GO TO 170
125   IF( TYP(IT).NE.HC ) GO TO 130
      CALL HCURVE(IT,LOC(I),DIFLOC1,DIFX1,DIFY1)
      GO TO 170
```

```
130   IF( TYP(IT).NE.SPIR ) GO TO 135
      CALL SPIRAL(IT,LOC(I),DIFLOC1,DIFX1,DIFY1)
      GO TO 170
135   WRITE(108,123)TYP(I)
123   FORMAT(1HO,'ERROR /1230 DIDN''T HAVE VC OR VT ',
     1'   THE ENTRY WAS ',A4)
      GO TO 999
170   NEWLOC = NEWLOC + DIFLOC1
      TOTY = TOTY + DIFY1
      TOTX = TOTX + DIFX1
      TOTDIST = TOTDIST + DIFLOC1
      WRITE(108,72) SYS,NUM,LOC(I),TYP(I),TOTX,TOTY
      GO TO 100
200   IT = I - 1
      IF( I.EQ.1 ) IT = TDIM
      IF( TYP(I).NE.II ) GO TO 205
      READ(109,6,END=900) SYS,NUM,LOC(I),TYP(I),
     1 NS(I),(DEG(I,K),K=1,3),EW(I),AHSTA
205   IF( TYP(IT).NE.HT ) GO TO 225
      CALL HTAN(IT,ENDLOC,DIFLOC1,DIFX1,DIFY1)
      GO TO 270
225   IF( TYP(IT).NE.HC ) GO TO 230
      CALL HCURVE(IT,ENDLOC,DIFLOC1,DIFX1,DIFY1)
      GO TO 270
230   IF( TYP(IT).NE.SPIR ) GO TO 235
      CALL SPIRAL(IT,ENDLOC,DIFLOC1,DIFX1,DIFY1)
      GO TO 270
235   WRITE(108,223)TYP(I)
223   FORMAT(1HO,'ERROR /2230 DIDN''T HAVE VC OR VT ',
     1'   THE ENTRY WAS ',A4)
      GO TO 999
270   TOTY = TOTY + DIFY1
      TOTX = TOTX + DIFX1
      TOTDIST = TOTDIST + DIFLOC1
      NEWLOC = NEWLOC + DIFLOC1
      IF( ABS(NEWLOC-ENDLOC).GT. .001) GO TO 910
226   IF( (SYS.NE.STSYS).AND.(NUM.NE.STNUM) ) GO TO 908
      DISTANCE = DISTANCE + TOTDIST
      DISTX    = DISTX    + TOTX
      DISTY    = DISTY    + TOTY
      WRITE(108,210)STSYS,STNUM,STLOC,ENDLOC,TOTDIST,TOTX,TOTY
210   FORMAT(//,' ROUTE NUMO ',A1,I3,3X,'START LOCO',
     1F8.1,3X,'END LOCO',F8.1,/,' DISTANCE TRAVELEDO',
     2F15.4,2X,'FEET',/,' DISTANCE IN XO',F15.4,2X,
     3'FEET',/,' DISTANCE IN YO',F15.4,2X,'FEET',/,1H1)
      REWIND 109
      GO TO 2
900   WRITE(108,901)MERR,SYS,NUM,LOC(I),TYP(I)
901   FORMAT(1HO,'ERROR /',I3,'O EOF ENCOUNTERED PREMATURELY'
     1,/,1H ,'LAST RECORD READO ',
     2A1,I3,F8.1,A2,F7.1,F6.2)
      GO TO 999
```

```
908    WRITE(108,909) MERR,STSYS,SYS,STNUM,NUM
909    FORMAT(1HO,'ERROR /',I3,'0 START SYSTEM',A1,2X,
      1'FINAL SYSTEMO ',A1,/,1H ,
      2'START NUMBERO ',I3,3X,'FINAL NUMBERO ',I3)
       GO TO 999
910    WRITE(108,911) MERR,NEWLOC,ENDLOC
911    FORMAT(1HO,'ERROR /',I3,'0 CALC LOCO ',F10.1,5X,
      1'BEYOND END LOCO ',F10.1)
999    STOP 'ABORTED IN MAIN LINE'
966    WRITE(108,967) DISTANCE,DISTX,DISTY
967    FORMAT(//,' TOTAL DISTANCE TRAVELEDO',F15.4,2X,
      1'FEET',/,' TOTAL DISTANCE IN XO',F15.4,2X,
      2'FEET',/,' TOTAL DISTANCE IN YO',F15.4,2X,'FEET')
       STOP'NORMAL END IN MAIN LINE'
C..
C..
       SUBROUTINE HTAN(L,ENDSTA,DIFSTA,DIFX,DIFY)
       CALL AZIMCALC(L,B2)
10     IF( AZIMADJ.EQ.0 ) GO TO 100
11     IF( ABS(AZIMUTH-B2).GT..001 ) GO TO 900
12     LT = L - 1
       LT1 = L + 1
       IF(L.EQ.TDIM) LT1 = 1
       IF(L.EQ.1) LT = TDIM
       DIFSTA = ENDSTA - LOC(L)
       X = 360.0 - (B2-90.0)
       X = X * PI /180.0
       DIFX = DIFSTA * COS(X)
       DIFY = DIFSTA * SIN(X)
20     IF( ABS(DIFSTA-(LOC(LT1)-LOC(L))).GT. .001) GO TO 902
       RETURN
C..    B2 IS TRUE AZIMUTH
100    IF( ABS(AZIMUTH-B2).LT. .001 ) GO TO 200
C..    ASSUMED AZIMUTH IS WRONG
       DIFAZIM = B2 - AZIMUTH
       DIFAZIMR = DIFAZIM * PI / 180.0
       XT1 = TOTX * COS( DIFAZIMR )
       YT1 = TOTX * SIN( DIFAZIMR )
       ANG = DIFAZIMR + PI / 2.0
       XT2 = TOTY * COS(ANG)
       YT2 = TOTY * SIN(ANG)
       TOTX = XT1 + XT2
       TOTY = YT1 + YT2
200    AZIMUTH = B2
       AZIMADJ = 99.99
       GO TO 12
900    WRITE(108,901)AZIMUTH,B2,NS(L),(DEG(L,K),K=1,3),EW(L)
901    FORMAT(1HO,'ERROR HTAN/110 CALC AZIMUTH   ',F15.4
      1,5X,'NE TANGENT AZIMUTH',F15.4,3X,'CALC FROM TAN DIR'
      23X,A1,3X,3I5,3X,A1)
       GO TO 12
902    WRITE(108,903)DIFSTA,LOC(LT1),LOC(L)
```

```
903   FORMAT(1HO,'ERROR HTAN/200 LENGTH CALC =',F15.4
     1,5X,'NEXT LOC',F14.4,5X,'PREV LOC=',F15.4)
      STOP '$ ABORTED IN HTAN SUBROUTINE $'
C..
C..
      SUBROUTINE HCURVE(J,ENDSTAC,DIFSTAC,DIFXC,DIFYC)
      J1 = J + 1
      IF( J.EQ.TDIM ) J1 = 1
      XLC = LOC(J1) - LOC(J)
      DELTA=DEG(J,1)+DEG(J,2)/60.0+DEG(J,3)/(60.0*60.0)
      DELTAR = DELTA * PI / 180.0
      RADIUS = XLC / DELTAR
      DIFSTAC = ENDSTAC - LOC(J)
      ALPHAR = DIFSTAC / RADIUS
      ALPHA = (ALPHAR*180.0) / PI
      DIST = 2*RADIUS*SIN(.5*ALPHAR)
      IF( EW(J).EQ.RIGHT ) GO TO 100
10    IF( EW(J).NE.LEFT ) GO TO 900
      CAZIM = AZIMUTH - ALPHA / 2.0
      IF( CAZIM.LT.0.0 ) CAZIM = 360.0 + CAZIM
      IF( DIFSTAC.EQ.0.0 ) ALPHA = + DELTA
      AZIMUTH = AZIMUTH - ALPHA
      IF( AZIMUTH .LT. 0.0 ) AZIMUTH = 360.0 + AZIMUTH
      GO TO 120
100   CAZIM = AZIMUTH + ALPHA / 2.0
      IF( CAZIM   .GT. 360.0 )   CAZIM   = -360.0 + CAZIM
      IF( DIFSTAC.EQ.0.0 ) ALPHA = + DELTA
      AZIMUTH = AZIMUTH + ALPHA
      IF( AZIMUTH .GT. 360.0 )   AZIMUTH = -360.0 + AZIMUTH
120   CAZIM = 360.0 - (CAZIM - 90.0)
      CAZIMR = (CAZIM*PI)/180.0
      DIFXC = DIST * COS(CAZIMR)
      DIFYC = DIST * SIN (CAZIMR)
      RETURN
900   WRITE(108,901) EW(J)
901   FORMAT(1HO,'ERROR HCURVE/100 WRONG CODING FOR '
     1,'CURVE DATA   SHOULD BE RIGHT OR LEFT CURVE BUT'
     2,' CODED',A1)
      STOP '$ ABORTED IN HCURVE SUBROUTINE$'
C..
C..
      SUBROUTINE AZIMCALC(K1,B4)
      B4=DEG(K1,1)+DEG(K1,2)/60.0+DEG(K1,3)/(60.0*60.0)
20    IF( (NS(K1).NE.NORTH).OR.(EW(K1).NE.WEST)  ) GO TO 30
      B4 = 360.0 - B4
      GO TO 100
30    IF( (NS(K1).NE.SOUTH).OR.(EW(K1).NE.WEST)  ) GO TO 40
      B4 = 180.0 + B4
      GO TO 100
40    IF( (NS(K1).NE.SOUTH).OR.(EW(K1).NE.EAST)  ) GO TO 10
      B4 = 180.0 - B4
      GO TO 100
```

```
10    IF( (NS(K1).NE.NORTH).OR.(EW(K1).NE.EAST)  ) GO TO 900
100   RETURN
900   WRITE(108,901)NS(K1),EW(K1)
901   FORMAT(1HO,'ERROR AZIMCALC/ O NO MATCH ON DIREC'
     1,'TIONS OF COMPASS   NS =',A1,5X,'EW =',A1)
902   STOP '$ ABORTED IN AZIMCALC SUBROUTINE$'
      SUBROUTINE SPIRAL(M,ENDSTAS,DIFSTAS,DIFXS,DIFYS)
      STOP'$ ERROR $ SPIRAL SUBROUTINE CALLED$'
999   RETURN
      END
```

The results of computer runs for both the horizontal and vertical data and the COGO runs for each route appear as follows:

```
RT NO.:P 11      STATION:   61423.90      START OF:C
X =      23.59     Y =       34.63
RT NO.:P 11      STATION:   61423.90      START OF:T
X =      23.59     Y =       34.63
RT NO.:P 11      STATION:   70074.37      START OF:C
X =    4896.29     Y =     7182.17
RT NO.:P 11      STATION:   70836.87      START OF:T
X =    5282.67     Y =     7838.87
RT NO.:P 11      STATION:   73500.00      START OF:C
X =    6477.53     Y =    10218.90
RT NO.:P 11      STATION:   73500.00      START OF:T
X =    6477.53     Y =    10218.90
RT NO.:P 11      STATION:   76903.37      START OF:C
X =    8038.93     Y =    13242.97
RT NO.:P 11      STATION:   79092.37      START OF:T
X =    9359.93     Y =    14974.41
RT NO.:P 11      STATION:   82035.87      START OF:C
X =   11525.20     Y =    16967.23
RT NO.:P 11      STATION:   85735.87      START OF:T
X =   14843.66     Y =    18453.38
RT NO.:P 11      STATION:   87794.69      START OF:C
X =   16892.56     Y =    18655.17
RT NO.:P 11      STATION:   88584.69      START OF:T
X =   17650.06     Y =    18865.24
RT NO.:P 11      STATION:   93022.81      START OF:C
X =   21660.01     Y =    20767.15


 ROUTE NUM: P 11    START LOC: 61382.0    END LOC: 94000.0
 DISTANCE TRAVELED:      32617.2500  FEET
 DISTANCE IN X:       22285.6523  FEET
 DISTANCE IN Y:       21482.2422  FEET
1
RT NO.:S476      STATION:    1098.00      START OF:C
X =     631.74     Y =     -898.06
RT NO.:S476      STATION:    2447.70      START OF:T
X =     971.15     Y =    -2175.52
RT NO.:S476      STATION:    3218.10      START OF:C
X =     899.10     Y =    -2942.55
RT NO.:S476      STATION:    3776.40      START OF:T
X =     928.29     Y =    -3498.09


 ROUTE NUM: S476    START LOC:       .0    END LOC:   4129.4
 DISTANCE TRAVELED:       4129.3984  FEET
 DISTANCE IN X:         997.9619  FEET
 DISTANCE IN Y:       -3844.1462  FEET
1
```

```
 RT NO.:S540      STATION:   42309.00      START OF:C
 X =   -2184.01     Y =   -1404.80
 RT NO.:S540      STATION:   43062.30      START OF:T
 X =   -2643.71     Y =   -1984.45
 RT NO.:S540      STATION:   43378.90      START OF:C
 X =   -2749.83     Y =   -2282.73
 RT NO.:S540      STATION:   44564.70      START OF:T
 X =   -3540.05     Y =   -3121.29
 RT NO.:S540      STATION:   44706.30      START OF:C
 X =   -3670.41     Y =   -3176.58

 RT NO.:S540      STATION:   45886.90      START OF:T
 X =   -4551.39     Y =   -3934.12
 RT NO.:S540      STATION:   12829.10      START OF:C
 X =  -11221.26     Y =  -14775.83
 RT NO.:S540      STATION:   13188.30      START OF:T
 X =  -11428.14     Y =  -15069.18
 RT NO.:S540      STATION:   19184.90      START OF:C
 X =  -15184.27     Y =  -19743.65
 RT NO.:S540      STATION:   20177.90      START OF:T
 X =  -16046.20     Y =  -20171.36


 ROUTE NUM: S540   START LOC: 39712.2   END LOC: 23835.8
 DISTANCE TRAVELED:      29910.5000  FEET
 DISTANCE IN X:      -19702.7305  FEET
 DISTANCE IN Y:      -20271.3750  FEET
1
 RT NO.:S362      STATION:     889.00      START OF:C
 X =     106.55     Y =     882.59
 RT NO.:S362      STATION:    3786.90      START OF:T
 X =   -1648.32     Y =    2710.70
 RT NO.:S362      STATION:    4952.90      START OF:C
 X =   -2810.66     Y =    2618.37
 RT NO.:S362      STATION:    5905.50      START OF:T
 X =   -3715.96     Y =    2852.70


 ROUTE NUM: S362   START LOC:       .0   END LOC:   6131.6
 DISTANCE TRAVELED:       6169.7031  FEET
 DISTANCE IN X:       -3936.1077  FEET
 DISTANCE IN Y:        2998.7825  FEET
1


 TOTAL DISTANCE TRAVELED:      72826.8125  FEET
 TOTAL DISTANCE IN X:        -355.2249  FEET
 TOTAL DISTANCE IN Y:         365.5012  FEET

!
```

Vertical Program, All Routes

```
ROUTE NUM: P 11    START LOC: 61382.0    END LOC: 94000.0
DISTANCE TRAVELED:       32617.2500  FEET
DIFFERENCE IN ELEVATION:        -42.9379  FEET

ROUTE NUM: S476    START LOC:      .0    END LOC:  4129.4
DISTANCE TRAVELED:        4129.3984  FEET
DIFFERENCE IN ELEVATION:          1.3970  FEET

ROUTE NUM: S540    START LOC: 39712.2    END LOC: 23835.8
DISTANCE TRAVELED:       29910.5000  FEET
DIFFERENCE IN ELEVATION:        116.2731  FEET

ROUTE NUM: S362    START LOC:      .0    END LOC:  6131.6
DISTANCE TRAVELED:        6169.7031  FEET
DIFFERENCE IN ELEVATION:        -75.8777  FEET


TOTAL DISTANCE TRAVELED:      72826.8125  FEET
TOTAL DIFFERENCE IN ELEVATION:         -1.1454  FEET

!
```

COGO Program, FAP11

```
?DUMP          1  20
       1                    .0000                    .0000

STORE          1                .0000                .0000
       2                   34.6273                  23.5916

STORE          2                34.6273              23.5916
       3                 7182.2079                4896.2953

STORE          3              7182.2079            4896.2953
       4                 7497.6749                5111.3578

STORE          4              7497.6749            5111.3578
       5                 7838.8887                5282.6597
STORE          5              7838.8887            5282.6597
       6                10218.8956                6477.5107
STORE          6             10218.8956            6477.5107
       7                13242.9884                8038.9185
STORE          7             13242.9884            8038.9185
       8                14225.6365                8546.2820

STORE          8             14225.6365            8546.2820
       9                14974.5487                9360.0051

STORE          9             14974.5487            9360.0051
      10                16967.4043                11525.3223

STORE         10             16967.4043            11525.3223
      11                18265.7942                12936.0749

STORE         11             18265.7942            12936.0749
      12                18453.7224                14844.1426

STORE         12             18453.7224            14844.1426
      13                18655.5200                16893.0289

STORE         13             18655.5200            16893.0289
      14                18694.6289                17290.1076

STORE         14             18694.6289            17290.1076
      15                18865.6167                17650.6131

STORE         15             18865.6167            17650.6131
      16                20767.5241                21660.5358
STORE         16             20767.5241            21660.5358
      17                21259.4265                22503.6390

STORE         17             21259.4265            22503.6390
      20                25944.7075                19204.9965

STORE         20             25944.7075            19204.9965
?
BREAK AT LOC 7READ+.9
```

```
?DUMP                1 8
       1                    .0000                    .0000

STORE                1          .0000                .0000
       2                   -898.0528                631.7477

STORE                2      -898.0528                631.7477
       3                   -1474.2625               1037.0904

STORE                3     -1474.2625               1037.0904
       4                   -2175.6744                971.1992
STORE                4     -2175.6744                971.1992
       5                   -2942.6974                899.1444

STORE                5     -2942.6974                899.1444
       6                   -3222.6647                872.8440

STORE                6     -3222.6647                872.8440
       7                   -3498.3332                928.3451

STORE                7     -3498.3332                928.3451
       8                   -3844.3892                998.0175

STORE                8     -3844.3892                998.0175
?END/OF/RUN
 *EXIT*
```

```
?DUMP          1  17
     1              .0000          .0000

STORE          1           .0000          .0000
     2          -1404.8025       -2184.0102

STORE          2      -1404.8025     -2184.0102
     3          -1616.3235       -2512.8564

STORE          3      -1616.3235     -2512.8564
     4          -1984.7061       -2643.9108

STORE          4      -1984.7061     -2643.9108
     5          -2282.9924       -2750.0281

STORE          5      -2282.9924     -2750.0281
     6          -2875.8905       -2960.9553

STORE          6      -2875.8905     -2960.9553
     7          -3121.6091       -3540.3005

STORE          7      -3121.6091     -3540.3005
     8          -3176.8987       -3670.6601

STORE          8      -3176.8987     -3670.6601
     9          -3415.0423       -4232.1453

STORE          9      -3415.0423     -4232.1453
    10          -3934.5106       -4551.7243

STORE         10      -3934.5106     -4551.7243
    11         -14776.2278      -11221.5935

STORE         11     -14776.2278    -11221.5935
    12         -14929.3683      -11315.8061

STORE         12     -14929.3683    -11315.8061
    13         -15069.5261      -11428.4288

STORE         13     -15069.5261    -11428.4288
    14         -19743.9969      -15184.5617

STORE         14     -19743.9969    -15184.5617
    15         -20157.2205      -15516.6042

STORE         15     -20157.2205    -15516.6042
    16         -20171.7135      -16046.5060

STORE         16     -20171.7135    -16046.5060
    17         -20271.7207      -19703.0387

STORE         17     -20271.7207    -19703.0387
?
```

COGO Program, FAS362

```
?DUMP                    1 8
       1                    .0000                    .0000

STORE                1       .0000                    .0000
       2                  882.5923                  106.5450

STORE                2       882.5923                 106.5450
       3                 2869.1696                   346.3609

STORE                3      2869.1696                 346.3609
       4                 2710.7223                  -1648.3560

STORE                4      2710.7223                -1648.3560
       5                 2618.3937                  -2810.6948

STORE                5      2618.3937                -2810.6948
       6                 2579.2135                  -3303.9411

STORE                6      2579.2135                -3303.9411
       7                 2852.7918                  -3716.2298

STORE                7      2852.7918                -3716.2298
       8                 2998.8698                  -3936.3726

STORE                8      2998.8698                -3936.3726
?END/OF/RUN
 *EXIT*
```

CHAPTER 2-VII

STORAGE AND RETRIEVAL OF VISUAL INFORMATION

Systems Reviewed

For the purposes of documentation the following hardware, as referred to in Chapter 1-VII, was conceptually reviewed:

Type	Hardware
Digitally-stored Images	IBM,4481 Film Reader-Recorder
Roll/sheet/aperture card	3M,200 Reader-Printer 3M,400 Reader-Printer 3M, Executive I Reader-Printer Kodak, Easamatic Reader Kodak, 310 Film Reader Kodak, Microstar Reader
"Keyed microfilm"	Kodak, Miracode II System
Central microfilm with remote access	Mosler, 410 Information Systems